PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-272454
(43)Date of publication of application : 08.10.1999

(51)Int.Cl. G06F 9/06

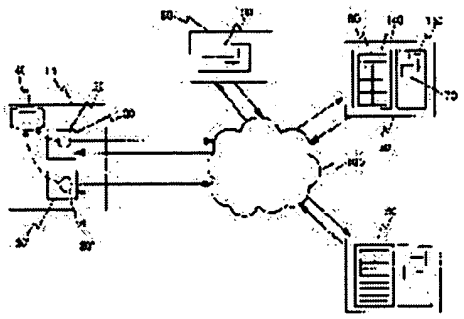
(21)Application number : 11-006203
(22)Date of filing : 13.01.1999

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : DONOHUE SEAMUS

(30)Priority
Priority number : 98 9801661 Priority date : 28.01.1998 Priority country : GB

(54) DEVICE FOR DISTRIBUTING SOFTWARE THROUGH COMPUTER NETWORK

(57)Abstract:
PROBLEM TO BE SOLVED: To considerably reduce cost and labor on a software distributor by software-updating a computer program through the use of a retrieved software resource.
SOLUTION: When an up data component 20 is executed, a product identifier and a product version release number, which are obtained at the time of introduction, are supplied to a search engine 90 as search arguments. Then, search on update which can be used for a specified software product is started. The update component 20 accesses to a list 60 by using URL and down-loads a file containing the part of the list 60 of update on the specified product, which can be used, is down-loaded. Then, comparison is executed between the identifier and the release number of the software product which is introduced at present and update which is listed in a retrieved file and which can be used. The comparison decides a possible growing path from a present version to an update version.



LEGAL STATUS

[Date of request for examination]	29.07.1999
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3385590
[Date of registration]	10.01.2003
[Number of appeal against examiner's decision	

of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] It is made the updater component for using it, when updating one or more computer programs introduced on the computer system connected within the computer network. The means for starting access to one or more identifiable locations in said network in which the required software resource of said one or more was prepared, in order to search one or more required software resources, An updater component including the means for giving renewal of software to one of said the introduced computer programs using the software resource with which one or more were searched.
[Claim 2] In order to identify the updating resource of usable relation, the comparison with the renewal resource of software obtained from the identifiable location of said one or more and one or more computer programs introduced on said computer system is performed. The means for comparing the updating criteria of the prior definition memorized in the updating resource and said computer system of said usable relation, An updater component including the means for giving renewal of software which downloads a software resource automatically and meets the updating criteria of said prior definition according to claim 1.
[Claim 3] The updating criteria of said prior definition are an updater component including the definition of the range of updating by applicable software product user license which can be permitted according to claim 2.
[Claim 4] The means for giving said renewal of software is an updater component including the means for introducing the software resource of usable relation according to claim 2 or 3 according to the instruction in which computer reading for the installation which is a part of software resource downloaded in accordance with the updating criteria of said prior definition for updating is possible.
[Claim 5] It is an updater component given in any of claim 1 thru/or claim 4 which works as a search parameter for using it in order that, as for said product identifier, said search engine may identify a network location the information for identifying one or more locations is held with said updater component, including the product identifier of a computer program product, said updater and identifier are adapted so that said product identifier may be supplied to a search engine, and they are.
[Claim 6] Said updater component answers that said search engine identifies the network location on which the list of usable renewal resources of software is held, and downloads said list and the premise software product of said resource. Said list and a premise software product are compared with the computer program introduced on said computer system. The updater component according to claim 5 which is adapted so that updating to said premise software product may be requested, when updating to said premise software product is required.
[Claim 7] Said updater component is identifiable and an updater component given in any of claim 1 thru/or claim 6 including the instruction for registering said updater component by the repository which can be accessed with other updater components so that it can contact they are by the updater component of others [have the introductory instruction in which machine reading for introducing said updater component on computer system is possible, and / instruction / said / introductory / component / said / updater].

[Claim 8] Said updater component contains API minded when a complementary computer program requests that a current updater component updates the computer program of that. Are adapted so that the updating approach may be called, in order that said current updater component may answer an updating request and may update the computer program of that. Said current updater component When the computer program of that needs renewal of a premise computer program, The updater component according to claim 6 or 7 which is adapted so that the request by which system generating was carried out may be sent to the updater component of said premise computer program of the computer program of that.
[Claim 9] The means for giving said updating is an updater component given in any of claim 1 which is adapted so that the version which the software [finishing / installation] which introduces the correction and expansion software which correct the existing software [finishing / installation], and permutes the introduced software upgraded may be introduced thru/or claim 8 they are.
[Claim 10] Said computer program code is a computer program product containing the integrated updater component indicated by any of claim 1 for updating said computer program code thru/or claim 9 they are including the computer program code recorded on the record medium in which computer reading is possible.
[Claim 11] Said computer program code is a computer program product which follows for any of claim 1 for updating said computer program code thru/or claim 9 being, and contains an integrated updater component including the computer program code for REKODARU on the record medium in which computer reading is possible.
[Claim 12] The computer program introduced on the computer system connected within the computer network is made into the approach for updating automatically. The step which distributes the updater component for using it when updating said computer program to said computer system, The step which provides the 1st network location with the software resource in which the download for forming said computer program in a version [finishing / updating] from a current version is possible, When performing in said computer system, it is the step which is adapted so that said updater component may carry out. (a) Are identifiable from the information held with said updater component. Or the step which starts access to said accessible 1st network location by which said software resource is arranged with said updater component, (b) The step which downloads said software resource on said computer system, (c) Approach containing the step which updates said computer program from said current version to a version [finishing / said updating] using the downloaded software resource.
[Claim 13] The step which prepares the list which updating usable for said computer program can computer read is included in the identifiable 2nd network location from the information in said updater component. It is made the step which is adapted so that it may be carried out with said updater component before accessing said 1st network location. In order to discriminate the updating resource of usable relation from the step which starts access to said 2nd network location in order to search said list The step which reads said list and performs the comparison with listed usable updating and said computer program on said 1st computer system, The approach containing the step which identifies the updating resource of the usable relation for updating which compares the updating resource of said usable relation with the updating criteria in said updater component defined in advance, and meets said updating criteria according to claim 12.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the device for accessing the information on a software function consolidation, correction, or an upgrade product through distributing software through a computer network, and a computer network. The "networks" of a computer may be a number of arbitration of computers which can exchange information mutually, or is arranged with the configuration of arbitration, and the connection method of arbitration is easy to be used for them.
[0002]
[Description of the Prior Art] Usually, software is distributed in the form of the program recorded on the diskette or a record medium like a compact disc. A customer buys the license which uses the software recorded on a record medium and its medium, and introduces the software on his own computer from the record medium after an appropriate time. The manufacture and distribution of a record medium which were recorded in advance are expensive, and the customer will be burdened with this cost. Moreover, it is not desirable that a customer tries hard that software should be placed an order for or purchased.
[0003] Since almost all software is frequently updated in order to correct a bug, and in order to add a new characteristic function after the software is sent to a user, especially the distribution cost is a problem. A certain type of software product is updated repeatedly every year. also although kicked, a thing for which it desires for the cost which sends a diskette or CD new to all registration members whenever software is upgraded or corrected to be expensive in prohibition, and for many customers to be the things of the newest [software / one's] moreover, and to be the version of the highest performance, and to be that errorless and from which all customers receive all updating is not desired. For example, requiring the upgrade of other premise software products which do not want the customer to impose many things to updating rather than what wants the customer to use a vendor, or to buy a high version, or shifting to a high version may need the migration of the data which make impossible the system of the customer between a certain periods.
[0004] Therefore, a software vendor releases the availability of the high version of its software, and has the inclination to leave the decision of whether to purchase the version to which the newest was upgraded to a customer. However, about a certain software product, it is suitable the upgraded version [as opposed to its own software product in a software vendor] or to send out an error correction and an functional enhancement code (known as "a patch") for the purpose of precedence study at least. If what kind of thing the polish of a specific enterprise is, remarkable cost and a remarkable effort will follow on the release of various these types of renewal of software.
[0005] Increasingly, the software negotiation contractor is going to use a certain software as a device for releasing the availability of updating to one's software, and use the Internet as a device of a distribution sake further. The Internet is the network of a computer network including the large-sized and small public and a private network without a single owner or a single controller, and any connected computers which run Internet Protocol software can receive

security control, and it can exchange other computers and information on arbitration which were connected to the Internet there. It does not depend on a single transmission medium for this compound aggregate of the network on which it has agreed with connecting mutually (for example, two-way communication may be produced through a satellite circuit, an optical fiber trunk line, the telephone line, a cable TV line, and a local wireless circuit).
[0006] World-Wide-Web Internet service (below, it is called a "web") is the broader-based information retrieval function in which access to much networks and accessible information is offered extremely, and the low cost communication link between computers by which the Internet connectivity was carried out can be offered. It is known that the customer of a software vendor who has Internet access will check a list for the available latest version of a product by manual operation, and will order those products on-line after an appropriate time. This decreases the amount of the document office work by which it is accompanied when placing an order for software (moreover, this is applicable like other products). A certain enterprise also enables its software to download from the website in a server computer directly to a customer's own computer (however). This download function by the reasons of security And since there is an inclination for applying a patch to need neither any modification to premise software nor what kind of data migration, It is limited to the demonstration copy or assessment copy of a bug restoration patch, a low cost program, and a program in many cases.
[0007] The information about World Wide Web Andrew Ford him -- the reference (International Thomson Publishing of London it will publish by the shrine in 1995) to twist and it was ["the spin of the web being carried out (Spinning the Web)"] entitled -- and -- John December and -- NeilRandall In the reference (SAMS Publishing of Indianapolis it will publish from a shrine in 1994) it was ["the released World Wide Web (The World Wide Web Unleashed)" to depend] entitled him -- It can see. For association with it, versatility, portability, and the ease of an activity, it is combined with an interactive multimedia presentation function, and the activity of WWW is growing at an explosive rate. WWW enables the computer of the arbitration which is connected to the Internet and has proper software and a proper hardware configuration to search with somewhere on the Internet the document of the arbitration made available.
[0008] Although the increment in such an activity of the Internet for the order of software and distribution saved the cost for a software vendor, it cannot be thoroughly depended for a vendor on all customers accessing suitable time amount to their web page about many software products, therefore its additional updating device is desirable.
[0009] In order to know whether it has the version best [of a software product] in a customer, and newest and release other than the problem of a manufacturing cost and the distribution cost relevant to a distribution medium, and in order to obtain updating and to give those updating, generally the problem that a customer needs to try hard for the purpose of a remarkable precedence study target exists. Although these efforts decrease when connection of the Internet is obtained, they are not that it is desirable for many users to even need the check aiming at precedence study of a website. This is because it is accompanied by measuring the version and release number in software [finishing / installation] and this list, in order to determine whether an order for it should be placed in order to determine [to navigate to the web page by which setting up notes for it checking, finding a software provider's website and accessing it, the newest software version, and the patch are listed, and] whether renewal of a related product is obtained. Even if the delay which is not desirable may exist in coming available because of placing an order for updating, and it activity in between and the updating can moreover download promptly, the task of shifting to the version to which the software product was upgraded will have a difficult thing. Updating becomes very redundant and these steps will become wasteful time, when it must be repeated for every system-software program installed on application, the control panel, the extension, the utility, and the system. therefore, the renewal of manual operation -- enough -- or there is an inclination which is not carried out regularly.
[0010] A software vendor has the problem of the relation which says that it does not know which version of its software is going to be used by each customer. even if the newest version of those software is energetically distributed by or online distribution of server control sending out CD -- to all the registered customers, there will still be no guarantee that a customer looks after right

installation of updating. This removes a certain freedom of a software-development person. Because, a software-development person is for having to make another concession to the user who has to maintain the version and reverse compatibility in front of his software, or generally does not upgrade.
[0011] It is known for the client/server computing environment that the system administrator by the side of a server will burden the end user in a client system with the high version of a software product by decision of him. However, this was possible when it had an access control for an administrator to update the system of a client. This does not take into consideration the user who does not want an upgrade to be imposed.
[0012] As a problem of the further relation, a software product may be said [needing other software products in many cases and], in order to enable it to work. For example, generally specification carries out the operating system pair of the application program, and it is written. Since the specific version of one product needs the specific version of other products in many cases, the 1st product may stop committing upgrading the 1st product, without upgrading other products as the result.
[0013] "Renewal 2.0 (Insider Updates 2.0) of The Insider" is an available software updater utility commercially from the The Insider software company (Insider Software Corporation). When the trigger of it is done by the user, it creates the inventory of the introduced software in a user's Apple Computer Mackintosh computer, compares this with the database of the available renewal patch of software (however, it is not the upgraded product version), and downloads renewal of relation. Although "renewal of The Insider" moves the charge for finding renewal of relation from a user to the maintenance side of a database, in order to limit access to an updating patch to connection with each database and to find updating, it needs doing one's best in the task which maintains the task which scans the Internet and an on-line service, and the database of available updating for the purpose of remarkable precedence study. "Renewal of The Insider" does not introduce updating, or does not correct a user's software by any approaches. "Renewal of The Insider" does not deal with the problem of the desynchronized premise software product.
[0014] The same product which does not download updating although the volume as which computer system was chosen is scanned and it connects with the database of the software title to Apple Computer Mackintosh in order to determine the introduced software is the "version master 1.5 (Version Master 1.5)" of a symmetry software company (Symmetry Software Corporation).
[0015] The another updating approach is offered by the "renewal server 1.1.6 (Shaman Update Server 1.1.6) of Sherman" from Sherman (Shaman Corporation). It consists of CD-ROM (it is updated every month and distributed) which a user introduces on a PowerMac file server, the client software for each Macintosh computer which inventory registration should be carried out and should be updated, and the means for accessing the FTP site which memorizes the library of current updating. "The renewal server of Sherman" creates the inventory of the computer by which network association was carried out, and downloads and distributes the latest version of software to each computer. A network administrator controls this inventory and an updating process mainly. Distribution of CD-ROM has the problem of the above-mentioned costs.
[0016]
[Problem(s) to be Solved by the Invention] According to the 1st aspect of affairs of this invention, the updater component for using it, when updating one or more computer programs introduced on the computer system connected within the computer network is offered. The updater component includes the means for giving renewal of software to one of said the introduced computer programs using the means for starting access to said one or more locations, and the software resource with which said one or more were searched, in order to search the information for identifying one or more locations in the network on which one or more required software resources are put, and the required software resource of said one or more.
[0017] As for the updater component by this invention, controlling automatically is desirable, without any dialogues by the user needing restoration of the bug in the upgrade of a related software product, and its product after the early acceptance about updating criteria. Updating criteria can be related with the licensing conditions of a product. This guarantees that the user

who chose the suitable updating polish can have the newest available software always most, when an error is automatically corrected from a user's viewpoint. Since an updater component processes this, a user does not need to learn the location which renewal of software produces, the method of obtaining them, or the approach of introducing them. In order to correct an error, it becomes unnecessary for a software vendor to ship special CD or a special diskette, and it becomes unnecessary to give the further description. A vendor can release a code easily on the basis of the increment in a target gradually so that popularity may be won without a customer doing his best in the description of a new product early.

[0018] The updater component by the desirable example of this invention performs the comparison between the available renewal of software and the introduced software in local computer system. It identifies whether it is connected with the software with which which was introduced, and renewal of available relation is compared with the updating criteria (these updating criteria are defined in advance to a current system or a current system user) held on local computer system. After an appropriate time, The renewal of software which satisfies the criteria defined in advance is downloaded automatically, and the updating is given.

[0019] Thus, giving renewal of software automatically is accompanied by introducing available software and the version patched and/or upgraded according to both the updating criteria defined in advance and the instruction for the installation downloaded with the program code which needs updating. The description of executing the downloaded instruction dynamically gives versatility in relation to the type of updating which can be processed with an updater component. It is usable also in order to make it possible to use a single general-purpose updater component with many various software products. As an option, the introductory instruction for a certain renewal of software may be encoded in advance within an updater component. Generally the "software resource" is the combination of a program code, the introductory instruction in which machine reading is possible, and required data modification of arbitration like address information.

[0020] The information for using it, when identifying a network location may be explicit network location information, or it may be the information on other arbitration usable as a search parameter for identifying a software vendor name or its location. In the desirable example, the information is a product identifier supplied to a search engine with an updater component, in order to start the search for identifying the network location of the relation which had the software resource for updating for the product memorized. This search is executable with the usual Internet (or other networks) search engine called with an updater component. When a search engine returns the identification information of a network location, when the list of renewal of available relation is searched from this location, that list is checked about the software product version and the updating criteria defined in advance held in local and those criteria are met, an updater component searches an updating resource and gives it on local computer system.

[0021] According to the desirable example of this invention, the stock-in-trade of standardized naming is used to the software resource which should form renewal of software, and, moreover, an updater component can be searched about these resources in the file system of a Network Operating System. This enables two or more locations to memorize a software resource, in order to ease a network availability problem, and it makes it easy that a developer and a distribution person offer the version to which those error restoration patches and a software product were upgraded. For example, a developer can make the new renewal of software obtained through the exhibited uniform resource locator (URL) which can be searched in order to use a known key word through the public network disk drive which used the known file name in those LANs.

[0022] As for an updater component, it is desirable that it is the indispensable part of the product which works in order that they may update. Therefore, an updater component is distributed to a software user with the initial version of a software product. After an appropriate time, an updater component obtains renewal of software automatically in accordance with presetting criteria (although renewal of a patch was received, you chose that a specific user received all updating like the mutual period of the successive searches about updating, or did it choose receiving only a certain updating of not receiving a permutation product version?), and

gives it.
[0023] As for the updating function of an updater component, it is desirable to include the updating itself. Before it searchs very much in quest of a software resource for an updater component to update the software product of the relation of that, in order to obtain updating to itself, updating criteria can be set so that a proper network location may always be accessed.
[0024] As for the updater component by this invention, it is desirable to include the means for confirming whether a premise product is available, and it is desirable for a required version to synchronize moreover as a part of the process which chooses the updating pass to a current product. In the desirable example, besides checking those availability, an updater component can be ordered to start updating to those software in the updater component relevant to a premise software product, when this is an updating polish [finishing / acceptance]. When the updater component of each software product can carry out the trigger of the updating to a premise product, updating can spread through the set of the introduced software product, without [without a user participates in the updating, or] needing to get to know the updating. This function is a big advantage compared with the conventional updater agent who does not deal with the problem of an asynchronous software version, when updating is performed, and in order to carry out the task for an end user, it supports the increasing trend in the software industry about the collaboration between distribution objects.
[0025] An updater component is that it is desirable to also include the device for inspecting the truth of the software which used and downloaded cryptographic algorithm. This makes unnecessary the software resource repository site which password protection of the dedication was carried out, and was protected by the option. A software resource may be anywhere on a network, as long as they can name correctly and are notified to a network search engine.
[0026] Therefore, this invention offers the agent and approach for giving the updating in order to obtain renewal of software. This approach reduces substantially the effort about the system administrator and end user who reduce substantially the cost and the effort about the software distributor which distributes and pursues renewal of software, and update for the introduced software.
[0027]
[Means for Solving the Problem] As shown in drawing 1 , the updater component 20 is introduced into the system memory of the usual computer system 10 by which network connection was carried out with the computer program 30 of relation. In order that the user of local computer system may introduce, the updater component may be distributed to a user by the storage (a diskette or CD), or may be clearly downloaded from other computer system. In the desirable example of this invention, an updater component is unified in the computer program (or apart from it, distributed to the program and coincidence to which they related through the same device) which had it meant that they performed maintenance. Since a user acquires it or is activated, an updater component is introduced as a part of introductory procedure of the program of the relation that it seems that it is unnecessary also in taking what kind of special action. The updater component [in / including / therefore / that the updater component registers itself with an operating system (an updater component, registering the central repository 40 which carried out **** distribution still more generally) / a local system] is identifiable by the address information in a register entry, and/or those product identifiers, and that of installation of each updater component is controllable. [at least]
[0028] The description of the desirable example of this invention is each updater component's being able to find other updater components which manage other software products, being found with other updater components, and being able to communicate with other updater components moreover. This function is used when it needs, in order that one updater component may update the updater component of another side on specific level before performing renewal of itself like the after-mentioned [the former]. This is made possible when an updater component registers into an operating system or other repositories 40.
[0029] In the desirable example, each registration entry includes two items, i.e., updater pass, and an updater network address. The pass is the location of an updater component binary file, as an updater component is risen by the operating system among a boot rise process. This

guarantees that it is ready for processing the request which an updater component is always active, and carried out the activity, or was generated from other updater components at it. A network address is the address used in order to find it on a network, and in order to communicate with it with the component on other computer system in a network.
[0030] The following naming agreement SoftwareVendorName+_product_name+_updater(s) to an updater component are used for the example of such registration that uses a UNIX (trademark) operating system and a TCP/IP protocol suite.
[0031] Pass registration is put in by UNIX/etc / inittab file in order to memorize a pass entry. For example, when the updater component to the DB2 (trademark) database product of IBM is introduced, it calls it the following forms, i.e., ibm_db2 pe_updater:2:respawn:/usr/abin/db2_updater_binary. The entry will be added to /etc / inittab file.

[0032] It will read this file, whenever computer system reboots, and the DB2 updater component will be started. When the "respawn" keyword in an updater entry causes a failure by a certain reason during system operation with a common updater component process, it guarantees that it is automatically restarted by the operating system. This approach will guarantee that all the updater components to all the introduced applications are always active.
[0033] Network location registration is put in by UNIX/etc / service file. For example, when a DB2 updater component is introduced, it is the following forms, i.e., ibm_db2 pe_updater. 5000/tcp It is called #net location of DB2 updater component. /etc/services The entry will be added to the file.
[0034] It will find it by searching this file about DB2 updater component name , when another updater component is wanted to communicate with a DB2 updater component. ibm_db2 pe_updater (actually carried out indirectly by UNIX call getservbyname().) The identifier is formed of a calling party according to standard naming agreement. It will get to know that the DB2 updater is hearing about the connection in a port number 5000, and will use the TCP protocol, when it is found. This enables it for the updater component in question to establish the link to a DB2 updater component, and to start conversation (for it to mention later about this).
[0035] In order for an updater component to find other updater components in other remote machines and to telephone to it The above-mentioned information is repository 40' (desirably) accessible [from both machines] and available for all the updater components that need it. Probably, it must be extended by having an accessible central database or an accessible distributed database also from where [in a web page or its network like a pan-network file]. An entry is the thing of the format of updater_name machine_ip_address (or DNA entry), a port number, and a protocol.
[0036] For example, the manufacturing department of a certain device may have three computer system with which a distributed software product works together mutually. Those systems are called a, b, and c. A web page or file manufacturing_collaborators.html Probably, the typical entry which can be set is as follows.
ibm_catia_updater a.manufacturing.com 5000 tcpibm_db2 pe_updater b.manufacturing.com 5100 tcpibm_cics_updater c.manufacturing.com 4780 tcp [0037] Next, an updater component can connect the updater component of other arbitration using the DNA name for creating an IP address, and the port number which the remote updater component is going to hear in the address, and can telephone to it.
[0038] Therefore, the step of the updater registration at the time of installation is as follows.
(1) /etc/inittab The entry in a file is created (an updater process code location is registered).
(2) /etc/service The entry in a file is created (an updater process local address is registered).
(3) Create the entry in a central database file (an updater process pan-network address is registered).
[0039] Supplying the local IP address of a web proxy server may also follow an introduction process on an updater component. Probably, it will be clear to this contractor for the registration practice of many alternatives to be possible.
[0040] An updater component includes the data field for the identifier to the software product with which they are related, and a version number. An updater component can be distributed to a

customer with these fields set to the null value, therefore in order that, as for an introductory procedure, an updater component may obtain an identifier, a current program version, and a release number, the initial step of questioning the software product of that is included. As an option, a software vendor can also code Product ID and the version number of relation in advance in an updater component.

[0041] That to which the system 10 of drawing 1 was connected in the network 100 of two or more remote server systems 50 and the computer containing 50' is shown. The software resource for updating to the program introduced on the local system 10 is obtained from the remote server system 50 and 50'. Each server system includes in storage the list 60 of the latest version of the software product obtained from the server, and the patch to the software product. Each vendor assumes that it is a thing using the software resource 70 required in order to assume that it is a thing using a website like the list 60 (that example is shown in drawing 2) of renewal of software which contains a product release history in the format which can be read and to form the release (this transition to new level from a software product release is called "growth pass" below) to new level from given level moreover with an updater component. The entry in the renewal list 60 of software contains the identification information 130 of the identification information 120 of a software resource required in order to give the updating, the premise software products of that, and those version numbers to each software product version 110. Resources required in a certain case are software and the perfect permutation version of an introductory instruction of relation. In the case of being another, a resource includes the patch cord for correcting the existing program (to for example, error correction sake), and an introductory instruction of a patch.

[0042] Although it will assume that a network 100 is the Internet about a current example, this invention can be carried out in the computer network of arbitration. The server system 80 is shown in a network 100, and the search engine 90 for using it, when finding the updating source location on a network is introduced into the server system. Although it is indicated that this separates and is installed from the local system 10, the need does not not necessarily exist. Each updater component 20 is related with the single program 30, and is shown by the drawing, and although it is moreover one description of this example of this invention to have the updater component of relation with which all the introduced software products manage them, it is not that each is essential to this invention of these descriptions so that it may mention later.

[0043] Next, the operation of an updater component will be explained with reference to drawing 3 , drawing 4 , and drawing 5 . When the introduced updater component is performed, it is supplying the 1st action of that to one or more search engines 90 by making into a search argument the product identifier and product version release number which were obtained at the time of installation, and starting the search about available updating to a specific software product (step 200). If a software vendor assumes that it is what supplies the list 60 of available renewal of a product referred to by the product identifier and the release number 110 through the website of these vendors, the search will identify the website 140 of relation where update information is acquired. Although the updater component will be tried so that a different search engine (you may be in a different geographical location to the 1st search engine) may be put into operation when the early attempt which is going to put a search engine into operation is unsuccessful, the period which it presetted may be retried waiting and after an appropriate time as an option. URL which identifies the website 140 of the relation to update information is returned to an updater component as a result of a search (step 210).

[0044] An updater component accesses a list 60 using the URL (step 220), and downloads the file 160 containing the part of the list 60 of available updating relevant to a specific product (step 230). After an appropriate time, an updater component carries out step 240-280 shown in drawing 4 and drawing 5 . Each file 160 contains the message digest (for example, MD5) signed in digital one. Next, the searched file 160 is analyzed using a digital signature check algorithm (it is (like the algorithm indicated by U.S. Pat. No. 5,231,668)) (step 240). This is important in order to check that a file 160 expresses the renewal list of right software to a specific software product, and that the file is not corrected after a sign. Furthermore, since these may contain two or more web pages URL in addition to a right thing, the check of a digital signature is the useful approach

of filtering the result of a search (a search may find other pages with the reference agreement to the product version containing the page which was not published by the software vendor which was able to name). A file is downloaded, and when the attempt which is going to verify it is not successful, an updater component progresses to the following URL found in the search.
[0045] Next, an updater component compares in local computer system between the identifier of a software product and release number by which current installation is carried out, and listed available updating in the searched file 160 (step 250). Although this comparison determines the possible growth pass from a current version to an updating version, these possible growth pass is compared with the updating criteria defined in advance after that (step 260), and any possible growth pass which does not meet the updating criteria is discarded. Therefore, it is determined that an updater component can apply a patch to a current version under whether it is possible to shift to an available high version from a current software product, and the license conditions by which current acceptance is carried out.
[0046] For example, the application of an available patch which the license of a software product may enable shift to what kind of future version about the product and application of what kind of available patch, may enable only the shift to the specified version, or corrects or corrects the error in a current version may only be permitted. The possible updating pass which cannot be used because of a limit of a current license is notified as a message of system generating sent to the software asset manager (an end user or IT procurement manager is sufficient) of the version by which current installation is carried out (step 270), and makes possible what they judge about whether a current license is enough.
[0047] Besides the license limit about updating which may take place, the updating criteria or the growth polish of an updater component Or [it should choose which of a cycle period (for example, for one week or one month), and two or more possible growth pass (whether the newest version permitted according to a license is always chosen)] or -- always choosing the newest patch and only notifying the availability of a high version -- it is -- making -- or -- or When premise software is already obtained in a local system, the criteria for determining whether it is made to only choose a high version are included. Growth criteria may also include control information -- when it should upgrade to the high version downloaded with an updater component. When shifting to a high software product version and data migration is required, this is outside operation time amount, or it is sometimes indispensable to be carried out every year only in every month or the single time amount by which the schedule was carried out, and this is controllable by the updater component.
[0048] The definition of a growth polish may contain the parameter which determines that renewal of a premise software product should be requested, when it is necessary to maintain the synchronization with the present product. About this, it will mention later in more detail. Probably, it will be clear to this contractor that there is big versatility in the criteria set up and applied with an updater component.
[0049] Next, an updater component determines specific growth pass (namely, version available in order to upgrade) from the set of possible growth pass using updating criteria (step 280). For example, an updater component may choose the possible version or possible release number of the highest of the available updating permitted by updating criteria, when it is an updating polish.
[0050] An updater component carries out the scan of the file system of an operating system, in order that a required software resource may already confirm whether it is available in local computer system (refer to drawing 3 (step 290), drawing 4 , and drawing 5). The required resource is a renewal work of software required in order to bring current application software to new level, and is renewal of software which needs to update premise software on required level. each updater component relevant to the product introduced [premise] -- (a) -- that it is introduced and (b) -- it is contacted in order that it may guarantee that it is in required premise level, or is in level higher than it (step 300). When all the required resources are obtained and checked in other local or machines, an updater component progresses to step 310 (refer to drawing 5) which forms the updated software version. When it is not checked, an updater component must obtain a required resource.
[0051] As shown in drawing 3 , drawing 4 , and drawing 5 , when the required software resource

for forming the updated version is not found in a local system, an updater component supplies the further request to one or more search engines, in order to find the required resource (step 320). A search engine returns one or more URL (step 330), and an updater component searches a software resource using these, and puts it into the storage of local computer system (steps 340 and 350). In this phase, an updater component or a user does not need to have at all information [/ what kind of correction or functional enhancement is contained in a high version]. It is determined which type of updating to be required for updating criteria so that the effort that a user investigates the content of all updating may be escaped. It is desirable actually for a user to be able to determine the effect of updating, therefore the software resource for updating includes the description about these effects that a user or an administrator can read.
[0052] For example, the software product which should be updated may be a word processor application program. When the sold word processor is not omitted in a certain font or does not contain a thesaurus, the patch for adding these features after that may be made available. An updater component has the function to add these to the word processor according to updating criteria.
[0053] The search about the required software resource which follows the initial search about an updating list in the another example of this invention is unnecessary (or when there is a high version to a premise software product and a patch, or the present product, it is only required.). Refer to following. This is because the updating software resource needed directly is memorized in relation to the list of required resources with the present product. That is, the list contains the pointer to the network location of the required resource so that it may be accompanied by selection of the pointer (pointer [further as opposed to the location of a premise software product depending on the case]) to the network location of updating [need / the growth pass from the list / to be chosen].
[0054] 2nd verification by the digital signature check is shortly carried out about the downloaded resource (refer to drawing 5). (step 360) After verifying the justification of the downloaded resource (step 360), an updater component performs installation in a target environment automatically according to an updating polish (step 310). Actually, although this may need the information or database activity parameter value from a user like a management password In the desirable example of this invention, installation of the downloaded code In the semantics that it does not need to obtain from the another place or a user gets to know a certain introductory information from the another place And when the updating criteria of a prior definition enable an updater component to update automatically, it is automatic in the semantics that generally it makes it possible for a user not to make any judgments at the time of activation.
[0055] Including the introductory instruction which was encoded in shell and in which machine reading is possible (for example, in the case of the application on interpretation language like Script or PERL or the Windows (trademark) operating system of Microsoft Corp. as the thing in which activation like setup.exe is possible) is known well. The updater component by this invention will download the instruction in which machine reading is possible with the software resource of relation (step 350), and will perform them automatically (step 310). Therefore, an updater component processes an introductory instruction automatically and the needed input from a man is avoided by the conventional approach. Script can be adapted so that the reuse of the information (for example, a user name, information like the password of an application administrator, an introductory directory, etc.) collected from the 1st introductory person which introduced the 1st version of an updater component may be carried out.
[0056] The approach of updating by the desirable example of this invention needs for a software vendor to compose a software resource with the need of forming, on other product level from one product level. For example, probably, generally the migration to a version 1.1.4 from a version 1.1.1 includes the required introductory sequence in the case of the ability for it to be nice and encode in a series of patches which should be given, and the introductory instruction in which machine processing is possible. Then, a user escapes the risk of peculiar efforts and a personal error by the approach which needs for a user to control the sequence of giving restoration and functional enhancement. Therefore, a software vendor deals with the problem how it should shift to another product level from one product level, instead of a customer, and an updater

component can move only to the level (namely, growth pass published by the software vendor to the specific existing product level) supported by the vendor.
[0057] An updater component generates a report (step 380) and it writes in a log report (step 390), and after an appropriate time, activation is ended until it activates again at the time of expiration of a predetermined updating cycle period (a repetitive period parameter is formed when updater component installation is carried out) (step 410) (an updater progresses to sleep or an idle state in the desirable (step 400) example).
[0058] The structure of the structure updater component of an A1. updater component consists of the general-purpose application programming interface to which the approach and other updater components for operating data and its data make it possible it, contact, and to communicate. Next, this structure will be explained in detail.
[0059] Updater component data: An updater component contains the following permanent data :P roduct_ID: Identifier of the software product managed with this updater component.
Current_Installed_Version: Version identifier (for example, version 3.1.0) to the introduced software.
Current_License: Version identifier corresponding to the software product version which a user can upgrade according to a current software license (for example, 4.0.z). As an option, this may be a license identifier (for example, LICI) for using it, when accessing the license conditions in which machine reading is possible.
Installation_Environment: A pair of list of an attribute name / attribute value. When an updater component is used for the first time, this is used with an updater component, in order to memorize the value into which it was put by the user. For example, updater installation user ID and a password or a root password, an introductory directory, the web proxy server address, a search engine URL, a log file name, the software asset manager electronic mail address, etc. The reuse of this data will be carried out when the subsequent renewal of automatic is needed.
Growth polish parameter: a.Growth_Cycle: Data which determine whether it must try in order that an updater component may update the software product of that every day every week or every month.
b. Growth_Type: Data which determine whether updating is restricted only to bug restoration and functional enhancement (namely, patch), or the upgrade to the latest release in each growth cycle is needed.
c. Force_Growth: (yes, no [/]) Parameter which determines whether other software resources should be forced so that it may upgrade when it is a premise for it upgrade of this software. (A certain practice will offer the control which is more flexible from this simple yes/Nor by forcing other software so that it may upgrade) .
Last_Growth_Time: A date and time amount when an updater component is performed at the end.

[0060] An updater component also contains the following un-permanent data :P ossible_Growth_Paths: Transient data showing available upgrade pass (for example, version number 3.1.d, 3.2.e, 4.0.a).
[0061] A2. private updater function: -- : in which updater component logic includes the following approaches -- Discover_Possible_Growth_Paths(): -- this software in the Internet (or other networks) sake Growth_Path Search about information. This search approach starts a search through a standard search engine server. The returned information is a still higher version and the premise product information on relation. Next, Growth_Path Information decreases according to a growth polish parameter. Growth_Path The check about whether the proper version of a premise product is obtained in a local computer and/or a remote computer is performed to all the members in a list. The updater component which manages these premise products is accessed, and growing up is forced when this is a polish. When all prior products exist locally on right level, or when it separates distantly, and is obtained on a network and it exists with a "compulsive growth" polish, moreover, i identifier to the higher version of a software product Possible_Growth_Paths It is added to a list.
Decide_Growth_Path(): Interpret a growth polish and choose single growth pass. A certain practice of this invention will be accompanied by a user dialogue choosing the pass, when a

consideration matter -- whether updating to other programs should be forced -- exists.
Get_Resources(Parameter: Chosen_Growth_Path): When Chosen_Growth_Path (for example, 3.2.0) is able to be given, search about a required resource (parameter Product_ID, Current_Installed_Version, Chosen_Growth_Path), and download all resources to a local computer. Probably, this contains the software which needs the introductory instruction in which a high version and machine processing are possible.
Install_Resources(): In order to hold introducing a required file into a right location or compiling those files, and software, process the introductory instruction which carries out the log of all the actions to a file (and the "uninstallation" approach is enabled to redo all actions) including correcting the existing structure of a system.
Grow(): -- :Discover_Possible_Growth_Paths() which starts an approach -- when possible growth pass does not exist, an updater component will be in an idle state -- if there is nothing as if, Decide_Growth_Path()Get_Resources(Parameter: Chosen_Growth_Path) Install_Resources().Grow() will write all completed actions in a log, and will end activation of an updater component. An updater component will be in an idle state until it is instructed to do so to the time amount which should be again checked about the new requirements for updating with other updater components.
[0062] An A3. general-purpose updater component API updater component contains the following general-purpose APIs. These functions could be called using the existing network communication software, such as a Remote Procedure Call, message orientation middleware, and ORB (object request broker).
Get_Release(): This function returns the release level of the product which is called with other updater components and managed with this updater component.
It is the product with which an updater component besides Update(new_level): calls this function, and is managed with this updater component new_level It moves to the new level expressed with parameter value. This is a private function. Grow() It calls.
Receive_Event (event details): When receiving a request so that an updater component may upgrade, it must tell the updater component of a call about when it completed updating. This function of the updater component requested in order that the updater component which carries out updating instead of other updater components might tell a success of that updating will be called. An event detail may be "Product ID, new release level and ok" or a string like "Product ID, new release level, and failure."
[0063] Automatic processing of the potential trouble of the desynchronized premise product by enabling enforcement of updating (or sending advice to a software asset manager, when enforcement of updating is not a part of updating polish) is an important advantage superior to the updating approach of the conventional technique.
[0064] The above-mentioned inspection (step 290) of the updater component registration database 40 and 40' is enabled to confirm whether the updating list file 160 which answered the initial search and was returned to the updater component is available at RIMOTO the information has local premise software and available, since the identifier 130 of premise software is included. Or it was local and was installed, when finding all the updater components installed by RIMOTO, it is clear that premise software is obtained, next in order to ensure that all premises are in right level, it is required to contact each updater component to each software product. Although it has a required product identifier to premise software 30', updater component 20' without a required version number is local, or when found in RIMOTO, and when enforcement of updating is an updating polish, the updater component 20 of the 1st computer program contacts this premise updater component 20' (step 300), and it requests that renewal of premise software product 30' of the relation of that is tried. This updater component 20' requests other updater components of the premise software of that to update those versions, when required.
[0065] In a certain phase, the updater component of relation is local, or in order to tell the requirements over a product new when not found in RIMOTO, in order to grow up a related product further, it is sent to an asset manager. When one updater component is not able to move to required level in a certain phase at the time of the chain of the updater request for making it grow up to be new level, this failure is reported to the updater component which it

called, and that updater component carries out the prompt directions of the failure to that renewal operation of a component etc. at the updater component which made the whole transaction start.
[0066] Therefore, an updater component can react to an external stimulus like the request from other updater components at everything but those automatic acts defined by those updating criteria.
[0067] B. The example of updating synchronization, next the example of the practice of the updating synchronization between two products will be explained. All products exist, and this example shows how to communicate with the updater component of another side so that one updater component may synchronize premise software, as it is in compatible release level.
[0068] CORBA(common object request broker architecture) ORB (object request broker) is used for the location between two updater components, and communication. When the above-mentioned general-purpose API is used, it is easy for a detailed person to develop a communication code on the technique of CORBA programming so that a certain updater component can talk with other updater components on a network. In this example, the updater component registration database 40 is the directory or folder including the file called "updater_component_name.iop" to the each introduced updater component obtained through a network (in addition, iop expresses an INTAOPE rubble object reference).
[0069] This file contains a series of cutting tools who may be changed into the criteria over that updater component by the updater component of the arbitration which reads a file for example, using a CORBA function, i.e., CORBA::Object::_string_to_object() in C++.
[0070] Furthermore, since it expresses the peculiar address to a corresponding updater component, these criteria can turn into criteria also to the updater component which is where on a network. When the updater component A makes the criteria over the updater component B, the updater component A is for example, C++ mapping A->Get_Release(). A general-purpose API function can be called by using it. It will return after that the value of the release level of the software managed with the updater component A.
[0071] In this example, the enquiry tool called two products, i.e., the DB2 product of IBM in machines M and N different, respectively, and an "enquiry builder" will be considered. (; which may be the machine with same Machines M and N -- as for this example, they only show that it may be separate) . As both products were outlined simply, it has the updater component which uses CORBAORB architecture. The ORB communication demon is active in the participating systems M and N.
[0072] Step 1. Registration phase: An operating system starts in System M and a DB2 updater component can be set to a Network File System folder or a directory. ibm_db2_updater.iop The file (based on a certain naming criterion used in order to help the subsequent search about the file) called is created promptly. This directory is made into a host in the machine of the arbitration which is not necessarily M or N. The file contains a series of usable cutting tools, in order to make the criteria over an updater component.
[0073] [Pseudo code]
Filehandle=open("/network/filesystem/directory",
"ibm_db2_updater.iop");ReferenceBytes=CORBA: :Object::_object_to_string();Write(FileHandle, ReferenceBytes);close(Filehandle); [0074] QueryBuilder An updater component starts, registration of that is written in the same directory or the same folder, and it is a file again in this case. ibm_querybuilder.iop It calls.
[0075] In this phase, both updater components are active and those existence and locations are registered into the network directory.
[0076] Step 2.QueryBuilder Although it is going to grow up to be a version 2 from a version 1, a premise is the DB2 version 2.1 or more than it. The following action sequence will arise. QueryBuilder It is expressed as QB and DB2 is expressed as DB2.
[0077] QB: Set to a network directory and it is a file. It searchs about ibm_db2_updater.iop (file name made according to the criterion). It finds a file, reads it and changes it into usable criteria.
[0078]
[Pseudo code]

if dbref =CORBA::Object::_string_to_object (readfile (ibm_db2_updater.iop)) then SUCCESS: We were connected to the updater. else FAIL: Premise software is colla tempestade borate Si. It does not exist in the set of a stem. SOFUTOUE It is ** about an electronic mail to an A asset manager. ** and a situation are notified.
The attempt which grows up to be an upgrade product is stopped.
endif [0079] Step 3. In this phase, we get to know that DB2 exists in somewhere in the set of the computer connected by network. Now, we need to get to know whether it is in right level. We are the above-mentioned general-purpose API functions out of QB updater. Get_Release() This is simply performed by performing. Therefore, QB updater is a client requested to a DB2 updater as telling what kind of release it is, as something is performed about it.
[0080] [Pseudo code]
db2_release = dbref->Get_Release(); [0081] For example, this returns a value "2.0."
[0082] Step 4 client side: QB updater component knows that this is not enough, and it needs a version 2.1. It is it. Force_Growth A parameter is investigated. In the parameter, before, as for "yes", it carries out the updating procedure of itself for premise software, it means that it must be made to grow up to required level. therefore, it waits for QB updater until carrying out that that is [****] right until it notifies to a DB2 updater that it grows up to a new release and premise software grows up to be a new release after an appropriate time goes wrong.
[0083]
[Pseudo code]
dbref->Update ("2.1" QBref); //QBref QB rise //They are the ready-made criteria over data. DB The trial for which//2 updater updates itself //When ** is ended, immediately, it or //The result of failure can be sent. //EVENT= nullWhile sent to a DB2 updater like (EVENT equals null) {; which nothing carries out} if (EVENT equals "SUCCESS") then [ ] -- this processor component (** -- Query) Software managed by Builder else which it is going to grow up Failure is written in a log.; You will not make it grow up.; ;endif. which progresses to sleep and is tried after that [0084] Server side: A DB2 updater component receives the request of growing up. It is tried so that it may grow up.
[0085] It reports a result to a call client (since it receives the criteria over a caller in a functional call, it learns how to contact a call client).
[0086]
[Pseudo code]
DB2 attempts to grow.if Growth Successful then QBRef->Receive_Event ("SUCCESS"); //Function Receive_ //Event Operation QB updater component stereo It can set to //-NENTO. EVENT The variable called The parameter sent in// API call //This SE of a value, i.e., IF statement, //When it is in a cushion "SUCCESS" ** //I want you to be cautious of setting.
else QBREF->Receive_Event("FAILURE");end if[0087] As mentioned above, the updating criteria defined in advance can determine which [of the available upgrade sets] to be applied, and which should be disregarded. Although they are identified noting that updating criteria have the available renewal of software, they can include an instruction in the updater component for there being no application of this updating in an updating polish, or sending advice to an end user or a system administrator, when impossible. Since one of the examples shown above is the polish which meant that an updating polish may need the upgrade of a premise software product, or the migration of data, and it, on the other hand, introduced a certain error correction patch (for example, when a software product is a database product) and it is obtained, I hear that it is not that the updating polish introduces the full permutation version of a software product, and there is. When upgrading one product needs the upgrade of the premise complementation product of another side, advice can be carried out rather than automatic installation of updating.
[0088] An updating polish can also determine extent of automation of an updating process by defining the environment where an updater requests the input from a user or an administrator.
[0089] Next, activation of the updater component of a specific example is carried out to explaining to a detail further. The function of this updater component is, maintaining the product [finishing / installation] called a "test" in the newest condition of having all released patches,

on the whole, and is not introducing the permutation version of a test. First, an updater component is constituted by the following data instantiation.
[0090]
Product ID: Test Version introduced [present]: 1.0.a The present license: LIC1 Introductory environment: "USERID:TestOwner, USERPASSWORD:easy"
"INSTALLPATH : /usr/bin/testapp/" Growth cycle: Week Growth type: Patch, the newest -- automatic Compulsive growth: Nothing The time of the last growth : 08/10/97 An updater is performed on Monday at 3:00 a.m. of the night of a day of the week every week, for example, every month, (a system administrator determines timing).
[0091] Below, the possible activation trace to the updater component of this example is shown.
[0092] Example step 1. of activation trace :>>>> which a growth cycle starts START : Discover_possible_Growth_Paths()* The search in a remote search engine (for example, Internet search engine) is performed using a phrase ("IBM Test 1.0.a Growth Paths"). A search returns URL released when a vendor outlined the current growth pass to a product.
* : which downloads URL -- a file content -- :"1.0.b and none; 2.0 and other_required product_product_id -- 1.0.c;"* A URL file is attested using a hash algorithm and a digital signature. When it is not Shinsei, it returns to the search about other URL adaptation criteria.
* growth_path_list It forms. : growth_path list = -- "1.0.c, none;2.0, and other_required_product_id -- 1.0.c;"* although all are removed -- patch level Growth_path Only what has the 2nd release number which matched the 1st version and 1.0 namely, -- increases from a list (based on Growth_Policy).
* growth_path list ="1.0.b and none;"* It guarantees that a premise exists to all the members in a list. In this example, all the members of a list agree ordinarily on these criteria.
* Candidate growth_paths Possible_Growth_Paths list = 1.0.b It puts in.
<<<< END : Discover_possible_Growth_Paths() [0093] Step 2. Next, an updater component is :>>>> START Decide_Growth_Path()* which follows the following processes through growth pass. A growth polish orders it for us to have to grow up to be the revision by which the newest was patched. (In this example, the newest revision usually comes out and a certain thing, i.e., it is 1.0.b, is determined)
* chosen_growth_path = 1.0.b<<<< END : Decide_Growth_Path() [0094] Step 3. Next, an updater component obtains the required resource for correcting current software level to new software level.
>>>> Get_Resources()* In a remote search engine (for example, Internet search engine), a search is performed using a phrase ("IBM Test REVISION 1.0.a to 1.0.b RESOURCES").
* A search returns URL, for example, the following.
ftp: //ftp.vendor-site/pub/test/resources/1.0.a-b"* An updater downloads the file directed by URL and puts it into the security-protection area where it inspects truth.
* An updater inspects truth (using the digital signature based on an RSA algorithm, or other approaches).
When a file is not Shinsei, it returns to a search (the caution 1 following reference).
* An updater unpacks a resource and uses it as a directory temporarily (refer to the following Note 2). These resources include the installation instruction (for example, instruction written by script language like a UNIX shell script or MVS REXX) in which machine processing is possible, and the file (binary compile or demand compile) which includes software fix actually.
<<<< END : Get_Resources() [0095] Caution caution 1- about the above-mentioned task In order to save time amount, an updater is called a "signature" and looks for the standard file containing URL before download of URL.
ftp://ftp.vendor-site/pub/test/resources/1.0.a-b And list of the content.
This is hashed and is signed. This signature is used, before download of the truth of URL (up to a certain range) of that, it is established promptly, and in order to check those resources after a directory unpacks the resource which the last downloaded temporarily, that information, i.e., a file list, can be used for an updater component. When the last URL downloads, it has truth checked again (in order to protect so that someone may not arrange a forgery object for a Shinsei URL location).

Note 2 - It is that an updater component investigates an introductory script, and the part of AMPAKKINGU corrects them based on the content of the introductory environmental data of that when required. For example, it is it when an introductory instruction is coded with a shell script. INSTALLPATH It is a token about all instances. "/usr/bin/testapp/" It will have and permute. Again, since the naming agreement of an attribute is the approach of the token substitution in an introductory instruction, it standardizes. On the whole, this enables automatic installation.
[0096] Step 4 Next, an updater component carries out a actual software upgrade.
>>>> START Install_Resources()* An introductory instruction is executed.
* : which updates the following values -- Current_Installed_Version = 1.0.bLast_Growth_Time = Date+Time* Before an upgrade expresses effectiveness, the electronic mail which tells [installation and] whether reboot of an operating system or restart of application is required is sent to a software asset manager.
<<<< END Install_Resources() [0097] This is termination of this present growth cycle. Seed is at present. Last_Growth_Time A value is updated and it ends after an appropriate time. The time amount spent for this cycle can become either to the number-of-cases time amount which the release new on the whole from a current thing will download, and will be introduced with new premise software from the number-of-cases second when the updater component knows that there is no upgrade pass to the version by which current installation is carried out.
[0098] The alternative over the above-mentioned example explained to the detail does not need an independent updater component to each different software product, but uses the single general-purpose updater component introduced on a system with the plug-in object and instruction peculiar to a product which are downloaded with each product. These objects carry out mutual coordination with a general-purpose code, and offer the same function of an updater component peculiar to the above-mentioned product. Probably, it will be clear to this contractor that this invention can be carried out in the system that not all the application programs introduced on the system but a certain application program and other software products have the updater component of relation, and for another modification to the above-mentioned example to be possible within the limits of [technical] this invention. [0099] As a conclusion, the following matters are indicated about the configuration of this invention.
[0100] (1) Make it the updater component for using it, when updating one or more computer programs introduced on the computer system connected within the computer network. The means for starting access to one or more identifiable locations in said network in which the required software resource of said one or more was prepared, in order to search one or more required software resources, An updater component including the means for giving renewal of software to one of said the introduced computer programs using the software resource with which one or more were searched.
(2) In order to identify the updating resource of usable relation, perform the comparison with the renewal resource of software obtained from the identifiable location of said one or more, and one or more computer programs introduced on said computer system. The means for comparing the updating criteria of the prior definition memorized in the updating resource and said computer system of said usable relation, An updater component given in the above (1) including the means for giving renewal of software which downloads a software resource automatically and meets the updating criteria of said prior definition.
(3) The updating criteria of said prior definition are an updater component given in the above (2) including the definition of the range of updating by applicable software product user license which can be permitted.
(4) The means for giving said renewal of software is an updater component given in the above (2) or the above (3) including the means for introducing the software resource of usable relation according to the instruction in which computer reading for the installation which is a part of software resource downloaded in accordance with the updating criteria of said prior definition for updating is possible.
(5) It is an updater component given in any of the above (1) thru/or the above (4) which works as a search parameter for using it in order that, as for said product identifier, said search engine

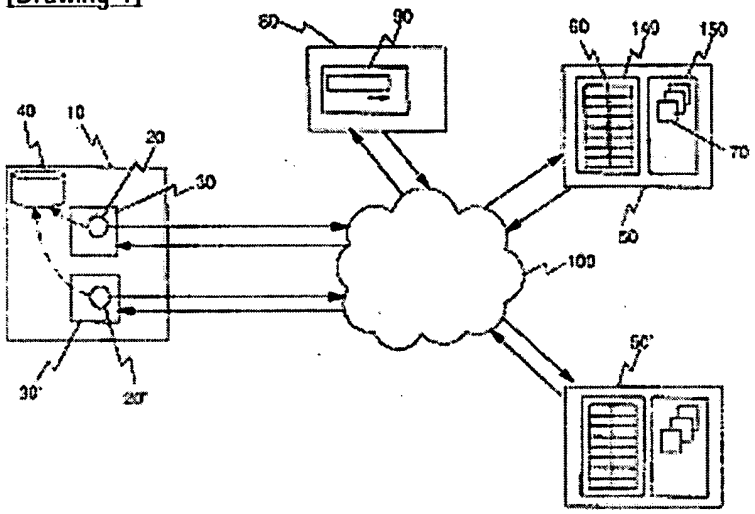
may identify a network location the information for identifying one or more locations is held with said updater component, including the product identifier of a computer program product, said updater and identifier are adapted so that said product identifier may be supplied to a search engine, and they are.
(6) Said updater component Answer that said search engine identifies the network location on which the list of usable renewal resources of software is held, and said list and the premise software product of said resource are downloaded. Said list and a premise software product are compared with the computer program introduced on said computer system. An updater component given in the above (5) which is adapted so that updating to said premise software product may be requested, when updating to said premise software product is required.
(7) Said updater component is identifiable and an updater component given in any of the above (1) thru/or the above (6) including the instruction for registering said updater component by the repository which can be accessed with other updater components so that it can contact they are by the updater component of others [have the introductory instruction in which machine reading for introducing said updater component on computer system is possible, and / instruction / said / introductory / component / said / updater].
(8) Said updater component API minded when a complementary computer program requests that a current updater component updates the computer program of that is included. Are adapted so that the updating approach may be called, in order that said current updater component may answer an updating request and may update the computer program of that. Said current updater component When the computer program of that needs renewal of a premise computer program, An updater component given in the above (6) or the above (7) which is adapted so that the request by which system generating was carried out may be sent to the updater component of said premise computer program of the computer program of that.
(9) The means for giving said updating is an updater component given in any of the above (1) which is adapted so that the version which the software [finishing / installation] which introduces the correction and expansion software which correct the existing software [finishing / installation], and permutes the introduced software upgraded may be introduced thru/or the above (8) they are.
(10) Said computer program code is the above (computer program product containing the integrated updater component indicated by 1 thru/or the above (any of 9 are they?).) for updating said computer program code including the computer program code recorded on the record medium in which computer reading is possible.
(11) Said computer program code is a computer program product which follows for any of the above (1) for updating said computer program code thru/or the above (9) being, and contains an integrated updater component including the computer program code for REKODARU on the record medium in which computer reading is possible.
(12) Make the computer program introduced on the computer system connected within the computer network into the approach for updating automatically. The step which distributes the updater component for using it when updating said computer program to said computer system, The step which provides the 1st network location with the software resource in which the download for forming said computer program in a version [finishing / updating] from a current version is possible, When performing in said computer system, it is the step which is adapted so that said updater component may carry out. (a) Are identifiable from the information held with said updater component. Or the step which starts access to said accessible 1st network location by which said software resource is arranged with said updater component, (b) The step which downloads said software resource on said computer system, (c) Approach containing the step which updates said computer program from said current version to a version [finishing / said updating] using the downloaded software resource.
(13) The step which prepares the list which updating usable for said computer program can computer read is included in the identifiable 2nd network location from the information in said updater component. It is made the step which is adapted so that it may be carried out with said updater component before accessing said 1st network location. In order to discriminate the updating resource of usable relation from the step which starts access to said 2nd network

location in order to search said list The step which reads said list and performs the comparison with listed usable updating and said computer program on said 1st computer system, An approach given in the above (12) containing the step which identifies the updating resource of the usable relation for updating which compares the updating resource of said usable relation with the updating criteria in said updater component defined in advance, and meets said updating criteria.

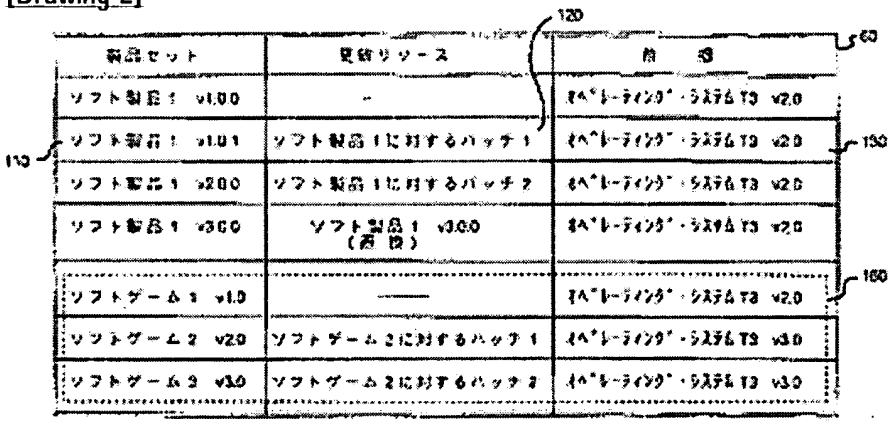
[Translation done.]

DRAWINGS

[Drawing 1]

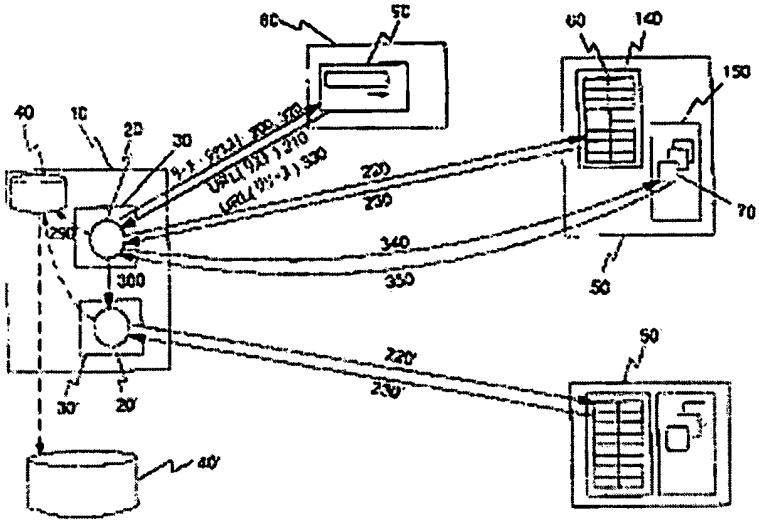


[Drawing 2]

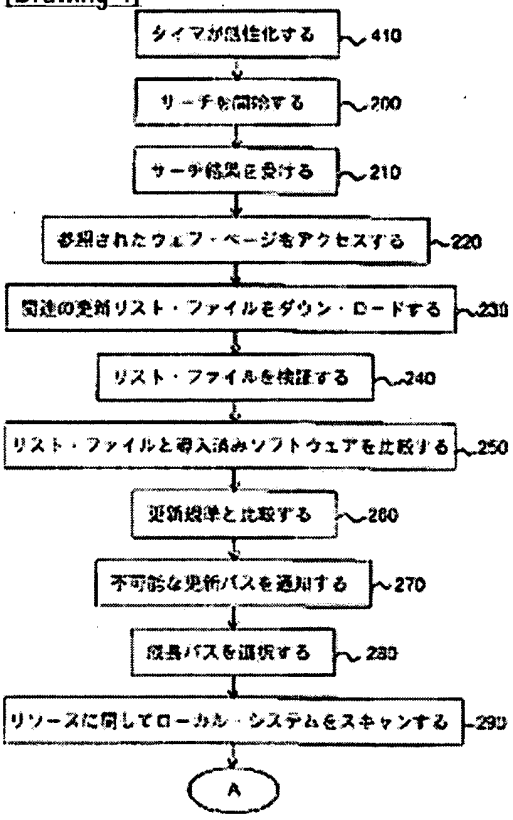


製品セット	更新リソース	前提
ソフト製品１ v1.00	-	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v2.0
ソフト製品１ v1.01	ソフト製品１に対するパッチ１	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v2.0
ソフト製品１ v2.00	ソフト製品１に対するパッチ２	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v2.0
ソフト製品１ v3.00	ソフト製品１ v3.00 (新規)	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v2.0
ソフトゲーム１ v1.0	——	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v2.0
ソフトゲーム２ v2.0	ソフトゲーム２に対するパッチ１	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v3.0
ソフトゲーム２ v3.0	ソフトゲーム２に対するパッチ２	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3 v3.0

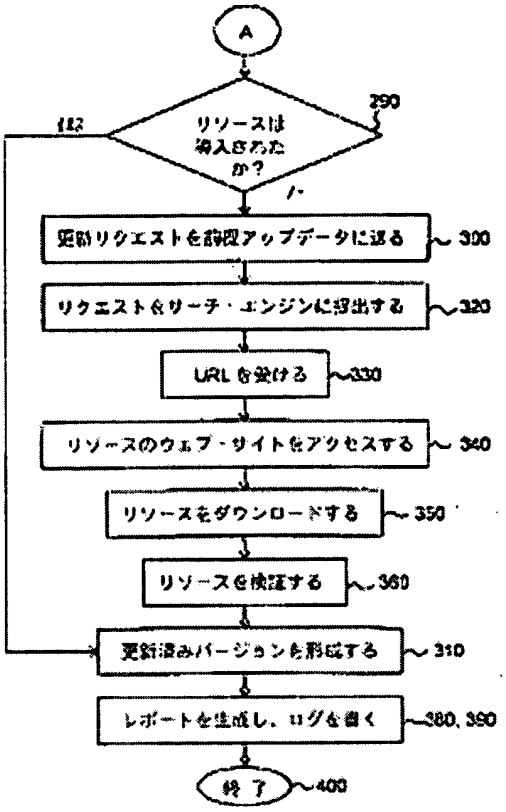
[Drawing 3]



[Drawing 4]



[Drawing 5]



[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－272454

(43)公開日　平成11年(1999)10月８日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/06	４１０	Ｇ０６Ｆ　9/06	４１０Ｑ

審査請求　未請求　請求項の数13　ＯＬ　（全 20 頁）

(21)出願番号　特願平11－6203

(22)出願日　平成11年(1999)１月13日

(31)優先権主張番号　９８０１６６１．１

(32)優先日　1998年１月28日

(33)優先権主張国　イギリス（ＧＢ）

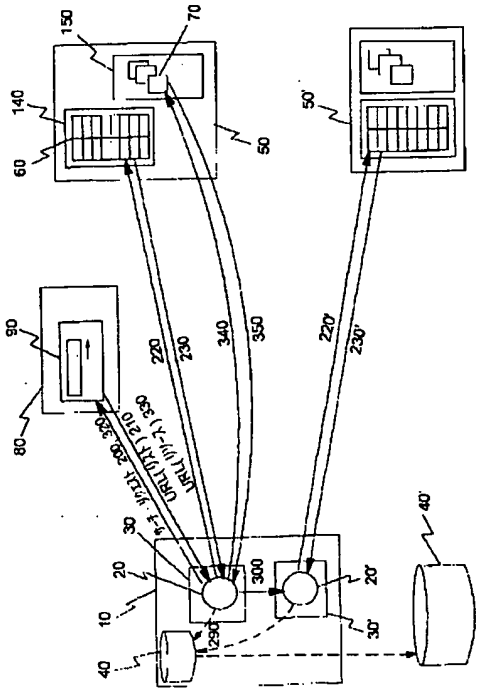
(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　シーマス・ドノヒュー
アイルランド共和国、ダブリン・５、アルタン、セント・ジョーンズ・コート　34

(74)代理人　弁理士　坂口　博　（外１名）

(54)【発明の名称】　コンピュータ・ネットワークを通してソフトウエア更新を配布する装置

(57)【要約】
【課題】コンピュータ・プログラムの更新を自動化するための方法及び機構を提供する。
【解決手段】通常、コンピュータ・プログラムは、ユーザが自分のコンピュータ・システム上に導入するために記録媒体を通して配布される。プログラムに関する補修、追加、及び新バージョンが行われる度に、ユーザがその更新を導入することを可能にするために、新しいＣＤ又はディスケットがユーザに配布される必要がある。最近では、或るソフドウエアはネットワークを介してダウンロード可能になったが、ユーザが更新情報を得てそれらを導入する労力、及びソフトウエア・ベンダが更新情報を分配するための労力は望ましくないままである。本発明は、コンピュータ・プログラムと関連付けられたアップデータ・エージェントを提供する。



1

【特許請求の範囲】

【請求項１】コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入された１つ又は複数のコンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントにして、

１つ又は複数の必要なソフトウエア・リソースを検索するために、前記１つ又は複数の必要なソフトウエア・リソースが設けられた前記ネットワーク内の１つ又は複数の識別可能なロケーションへのアクセスを開始するための手段と、

１つ又は複数の検索されたソフトウエア・リソースを使用して前記導入されたコンピュータ・プログラムの１つにソフトウエア更新を施すための手段と、

を含むアップデータ・コンポーネント。

【請求項２】使用可能な関連の更新リソースを識別するために、前記１つ又は複数の識別可能なロケーションから得られるソフトウエア更新リソースと前記コンピュータ・システム上に導入された１つ又は複数のコンピュータ・プログラムとの比較を行い、前記使用可能な関連の更新リソースと前記コンピュータ・システムにおいて記憶された事前定義の更新基準とを比較するための手段と、

ソフトウエア・リソースを自動的にダウンロードし、前記事前定義の更新基準を満たすソフトウエア更新を施すための手段と、

を含む請求項１に記載のアップデータ・コンポーネント。

【請求項３】前記事前定義の更新基準は適用可能なソフトウエア製品ユーザ・ライセンスによる許容し得る更新の範囲の定義を含む請求項２に記載のアップデータ・コンポーネント。

【請求項４】前記ソフトウエア更新を施すための手段は前記事前定義の更新基準に従って及び更新のためにダウンロードされたソフトウエア・リソースの一部である導入のためのコンピュータ読み取り可能な命令に従って、使用可能な関連のソフトウエア・リソースを導入するための手段を含む請求項２又は請求項３に記載のアップデータ・コンポーネント。

【請求項５】１つ又は複数のロケーションを識別するための情報が前記アップデータ・コンポーネントによって保持され、コンピュータ・プログラム製品の製品識別子を含み、

前記アップデータ・識別子は前記製品識別子をサーチ・エンジンに供給するように適応し、

前記製品識別子は前記サーチ・エンジンがネットワーク・ロケーションを識別するために使用するためのサーチ・パラメータとして働く、

請求項１乃至請求項４の何れかに記載のアップデータ・コンポーネント。

【請求項６】前記アップデータ・コンポーネントは、

2

使用可能なソフトウエア更新リソースのリストが保持されているネットワーク・ロケーションを前記サーチ・エンジンが識別することに応答して前記リスト及び前記リソースの前提ソフトウエア製品をダウンロードし、

前記リスト及び前提ソフトウエア製品と前記コンピュータ・システム上に導入されたコンピュータ・プログラムとを比較し、

前記前提ソフトウエア製品に対する更新が必要である場合に前記前提ソフトウエア製品に対する更新をリクエストするように適応する請求項５に記載のアップデータ・コンポーネント。

【請求項７】前記アップデータ・コンポーネントはコンピュータ・システム上に前記アップデータ・コンポーネントを導入するための機械読み取り可能な導入命令を有し、

前記導入命令は前記アップデータ・コンポーネントが他のアップデータ・コンポーネントによって識別可能及び接触可能であるように他のアップデータ・コンポーネントによってアクセスし得るリポジトリによって前記アップデータ・コンポーネントを登録するための命令を含む、

請求項１乃至請求項６の何れかに記載のアップデータ・コンポーネント。

【請求項８】前記アップデータ・コンポーネントは、現在のアップデータ・コンポーネントがそれのコンピュータ・プログラムを更新することを相補的なコンピュータ・プログラムがリクエストする時に介するＡＰＩを含み、

前記現在のアップデータ・コンポーネントは更新リクエストに応答してそれのコンピュータ・プログラムを更新するために更新方法を呼び出すように適応し、

前記現在のアップデータ・コンポーネントは、それのコンピュータ・プログラムが前提コンピュータ・プログラムの更新を必要とする時、システム発生されたリクエストをそれのコンピュータ・プログラムの前記前提コンピュータ・プログラムのアップデータ・コンポーネントに送るように適応する請求項６又は請求項７に記載のアップデータ・コンポーネント。

【請求項９】前記更新を施すための手段は存在する導入済みのソフトウエアを修正する訂正及び機能拡張ソフトウエアを導入し、

導入されたソフトウエアを置換する導入済みのソフトウエアのアップグレードしたバージョンを導入するように適応する請求項１乃至請求項８の何れかに記載のアップデータ・コンポーネント。

【請求項１０】コンピュータ読み取り可能な記録媒体上に記録されたコンピュータ・プログラム・コードを含み、

前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための請求項１乃

至請求項９の何れかに記載された統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。
【請求項１１】コンピュータ読み取り可能な記録媒体上のレコーダルのためのコンピュータ・プログラム・コードを含み、
前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための請求項１乃至請求項９の何れかに従って統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。
【請求項１２】コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入されたコンピュータ・プログラムを自動的に更新するための方法にして、
前記コンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントを前記コンピュータ・システムに配布するステップと、
前記コンピュータ・プログラムを現在のバージョンから更新済みのバージョンに形成するためのダウンロード可能なソフトウエア・リソースを第１ネットワーク・ロケーションに提供するステップと、
前記コンピュータ・システムにおいて実行される時、前記アップデータ・コンポーネントが遂行するように適応するステップであって、
（ａ）前記アップデータ・コンポーネントによって保持された情報から識別可能であり、又は前記アップデータ・コンポーネントによってアクセス可能であって、前記ソフトウエア・リソースが配置される前記第１ネットワーク・ロケーションへのアクセスを開始するステップと、
（ｂ）前記ソフトウエア・リソースを前記コンピュータ・システム上にダウンロードするステップと、
（ｃ）ダウンロードされたソフトウエア・リソースを使用して前記コンピュータ・プログラムを前記現在のバージョンから前記更新済みのバージョンに更新するステップと、
を含む方法。
【請求項１３】前記アップデータ・コンポーネントにおける情報から識別可能な第２ネットワーク・ロケーションに前記コンピュータ・プログラムにとって使用可能な更新のコンピュータ読み取り可能なリストを設けるステップを含み、
前記アップデータ・コンポーネントによって、前記第１ネットワーク・ロケーションをアクセスする前に遂行されるように適応するステップにして、
前記リストを検索するために前記第２ネットワーク・ロケーションへのアクセスを開始するステップと、
使用可能な関連の更新リソースを識別するために、前記リストを読み取り、リストされた使用可能な更新と前記第１コンピュータ・システム上の前記コンピュータ・プログラムとの比較を行うステップと、
前記使用可能な関連の更新リソースと前記アップデータ・コンポーネントにおける事前定義された更新基準とを比較し、前記更新基準を満たす更新のための使用可能な関連の更新リソースを識別するステップと、
を含む請求項１２に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、コンピュータ・ネットワークを介してソフトウエアを配布すること及びコンピュータ・ネットワークを介してソフトウエア機能強化、訂正、又は新バージョンの情報をアクセスするための機構に関するものである。コンピュータの「ネットワーク」は、相互に情報を交換することができる任意の数のコンピュータであってもよく、或いは任意の構成で配列され、任意の接続方法を使用するものでよい。
【０００２】
【従来の技術】通常、ソフトウエアは、ディスケット又はコンパクト・ディスクのような記録媒体上に記録されたプログラムの形式で配布されている。顧客は、記録媒体及びその媒体上に記録されたソフトウエアを使用するライセンスを買い、しかる後、その記録媒体から自分のコンピュータ上にそのソフトウエアを導入する。事前記録された記録媒体の製造及び配布は高価なものであり、このコストは顧客に課せられるであろう。又、顧客がソフトウエアを注文又は購入すべく努力することは望ましいことではない。
【０００３】ほとんどのソフトウエアは、そのソフトウエアがユーザに送達された後、バグを訂正するために及び新たな特徴的機能を追加するために頻繁に更新されるので、その配布コストが特に問題である。或るタイプのソフトウエア製品は毎年何回も更新される。ソフトウエアがアップグレードされ又は訂正される度にすべての登録メンバに新しいディスケット又はＣＤを送るコストは禁止的に高価であり、しかも、多くの顧客は自分のソフトウエアが最新のものであって最高パフォーマンスのバージョンであること、及びエラーのないものであることを望んでいるけれども、すべての顧客があらゆる更新を受けることを望んでいるわけではない。例えば、ベンダは顧客が使うことを望んでいるものよりも多くのものを更新に対して課することがあり、或いは、新しいバージョンは顧客が買うことを望んでいない他の前提ソフトウエア製品のアップグレードを要求することがあり、或いは、新しいバージョンに移行することは或る期間の間顧客のシステムを不能にするデータのマイグレーションを必要とすることがある。
【０００４】従って、ソフトウエア・ベンダは、自分のソフトウエアの新しいバージョンの可用性を公表し、最新のアップグレードされたバージョンを購入すべきかどうかの決定を顧客に委ねる傾向がある。しかし、或るソフトウエア製品に関しては、ソフトウエア・ベンダが、自分のソフトウエア製品に対するアップグレードされた

バージョン又は少なくともエラー訂正及び機能強化コード（「パッチ」として知られている）を先行学習を目的に送り出すことが適している。特定の企業のポリシがどのようなものであろうと、これらの種々なタイプのソフトウエア更新のリリースにはかなりのコストと労力が伴う。

【０００５】益々、ソフトウエア流通業者は、自分のソフトウエアに対する更新の可用性を公表するための機構として、更に或るソフトウエアを配布ための機構としてインターネットを使用しようとしている。インターネットは、単一のオーナ又はコントローラを持たない大型及び小型の公衆及び私設ネットワークを含むコンピュータ・ネットワークのネットワークであり、そこでは、インターネット・プロトコル・ソフトウエアを走らせる如何なる接続されたコンピュータもセキュリティ・コントロールを受け、インターネットに接続された他の任意のコンピュータと情報を交換することができる。相互に接続することに同意したネットワークのこの複合集合体は単一の伝送媒体に依存するのではない（例えば、双方向通信は衛星回線、光ファイバ中継回線、電話回線、ケーブル・テレビ線、及びローカル無線回線を介して生じ得る）。

【０００６】ワールド・ワイド・ウェブ・インターネット・サービス（以下では、「ウェブ」と呼ぶ）は、極めて多数のネットワーク・アクセス可能な情報へのアクセスを提供し、インターネット接続されたコンピュータ相互間の低コスト通信を提供し得る広域情報検索機能である。インターネット・アクセスを有するソフトウエア・ベンダの顧客が製品の入手可能な最新バージョンをリストを手操作でチェックし、しかる後、それらの製品をオンラインで注文することが知られている。これは、ソフトウエアを注文する場合に伴う書類事務作業の量を減少させる（しかも、これは他の製品に同様に適用可能である）。或る企業は自分のソフトウエアがサーバ・コンピュータにおけるウェブ・サイトから顧客自身のコンピュータに直接にダウンロードされることも可能にしている（しかし、このダウンロード機能は、セキュリティ上の理由で及びパッチを適用することが前提ソフトウエアに対する如何なる変更も又は如何なるデータ・マイグレーションも必要としない傾向があるため、バグ修復パッチ、低コスト・プログラム、及びプログラムのデモ・コピー又は評価コピーに限定されることが多い）。

【０００７】ワールド・ワイド・ウェブに関する情報は、Andrew Ford 氏による「ウェブをスピンする（Spinning the Web）」と題した文献（ロンドンの International Thomson Publishing 社により１９９５年発行）、及び John December 及び NeilRandall 氏による「解放されたワールド・ワイド・ウェブ（The World Wide Web Unleashed）」と題した文献（インディアナポリスの SAMS Publishing 社より１９９４年発行）において見ることができる。WWWの使用は、それと融通性、携帯性、及び使用の容易性との結合のため、対話式マルチメディア・プレゼンテーション機能と結合されて爆発的な速度で成長しつつある。WWWは、インターネットに接続され且つ適正なソフトウエア及びハードウエア構成を有する任意のコンピュータが、インターネット上のどこかで利用可能にされた任意のドキュメントを検索することを可能にする。

【０００８】ソフトウエアの注文及び配布のためのインターネットのこのような使用の増加はソフトウエア・ベンダのためのコストを節約したが、多くのソフトウエア製品に関して、ベンダは、自分のウェブ・ページを適切な時間にすべての顧客がアクセスするということに完全に依存することはできず、従って、付加的な更新機構が望ましい。

【０００９】製造コスト及び配布媒体と関連した配布コストの問題の他に、顧客がソフトウエア製品の最良の及び最新のバージョン及びリリースを有するかどうかを知るために、及び更新を得てそれらの更新を施すために、一般には、顧客はかなりの先行学習的を目的に努力を行う必要があるという問題が存在する。この努力は、インターネットの接続が得られる時には少なくなるけれども、ウェブ・サイトの先行学習を目的にしたチェックを必要とすることさえも多くのユーザにとって望ましいことではない。これは、それがチェックを行うための注意点を設定すること、ソフトウエア提供者のウェブ・サイトを見つけてアクセスすること、最新のソフトウエア・バージョン及びパッチがリストされているウェブ・ページにナビゲートするすること、及び関連製品の更新が得られるかどうかを決定するために及びそれが注文されるべきかどうかを決定するために、導入済みのソフトウエアとこのリスト内のバージョン及びリリース番号とを比較することを伴うためである。更新を注文することとそれが使用のために利用可能になることとの間に望ましくない遅延が存在することがあり、しかも、たとえその更新が直ちにダウンロードされ得るものであっても、ソフトウエア製品のアップグレードされたバージョンに移行するというタスクは困難であることがある。これらのステップは、アプリケーション、コントロール・パネル、拡張子、ユーティリティ、及びシステム上にインストールされたシステム・ソフトウエア・プログラム毎に繰り返されなければならない場合、更新は非常に冗長となり、時間浪費的なものになる。従って、手操作の更新は十分に又は規則的に遂行されない傾向がある。

【００１０】ソフトウエア・ベンダは、自分のソフトウエアのどのバージョンが各顧客によって使用されようとしているかを知らないと云う関連の問題がある。たとえ、それらのソフトウエアの最新のバージョンがすべての登録された顧客に精力的に（ＣＤを送出することによって又はサーバ制御のオンライン配布によって）配布さ

れたとしても、顧客が更新の正しい導入の面倒を見るという保証は依然としてない。これはソフトウエア開発者の或る自由を取り去る。なぜならば、ソフトウエア開発者は、一般に、自分のソフトウエアの前のバージョンと逆互換性を維持しなければならないか、或いはアップグレードしないユーザに対して別の譲歩を行わなければならないためである。

【００１１】クライアント／サーバ・コンピューティング環境では、サーバ側におけるシステム・アドミニストレータが自分の判断でソフトウエア製品の新しいバージョンをクライアント・システムにおけるエンド・ユーザに課することが知られている。しかし、これはアドミニストレータがクライアントのシステムを更新するためのアクセス制御を有する場合に可能であったにすぎない。これは、アップグレードが課せられることを望まないユーザを考慮するものではない。

【００１２】更なる関連の問題として、ソフトウエア製品は、それが働くことを可能にするために他のソフトウエア製品を必要とすることが多いということがある。例えば、アプリケーション・プログラムは、一般に、特定のオペレーティング・システム対して書かれる。１つの製品の特定のバージョンは他の製品の特定のバージョンを必要とすることが多いので、他の製品をアップグレードすることなく第１の製品をアップグレードすることは、その結果として第１の製品が働かなくなることがある。

【００１３】「インサイダ更新２．０（Insider Updates 2.0）」は、インサイダ・ソフトウエア社（Insider Software Corporation）から商業的に入手可能なソフトウエア・アップデータ・ユーティリティである。それは、ユーザによってトリガされる時、ユーザのアップル・マッキントッシュ・コンピュータにおける導入済みソフトウエアのインベントリを作成し、これと利用可能なソフトウエア更新パッチ（しかし、アップグレードされた製品バージョンではない）のデータベースとを比較し、関連の更新をダウンロードする。「インサイダ更新」は関連の更新を見つけるための責任をユーザからデータベースの保守側に移すが、更新パッチへのアクセスは個々のデータベースへの接続に限定され、更新を見つけるためにインターネット及びオンライン・サービスをスキャンするタスク及び利用可能な更新のデータベースを維持するタスクはかなりの先行学習を目的に努力することを必要とする。「インサイダ更新」は更新を導入せず、或いはユーザのソフトウエアを如何なる方法でも修正しない。「インサイダ更新」は非同期化された前提ソフトウエア製品の問題を処理しない。

【００１４】導入されたソフトウエアを決定するためにコンピュータ・システムの選択されたボリュームをスキャンし、アップル・マッキントッシュに対するソフトウエア・タイトルのデータベースに接続するが更新をダウンロードしない同様の製品は、シンメトリ・ソフトウエア社（Symmetry Software Corporation）の「バージョン・マスタ１．５（Version Master 1.5）」である。

【００１５】別の更新方法がシャーマン社（Shaman Corporation）からの「シャーマン更新サーバ１．１．６（Shaman Update Server 1.1.6）」によって提供される。それは、ユーザがＰｏｗｅｒＭａｃファイル・サーバ上に導入するＣＤ－ＲＯＭ（毎月更新され、配布される）と、インベントリ登録され、更新されるべき各マッキントッシュ・コンピュータのためのクライアント・ソフトウエアと、現在の更新のライブラリを記憶するＦＴＰサイトをアクセスするための手段とより成る。「シャーマン更新サーバ」は、ネットワーク結合されたコンピュータのインベントリを作成し、ソフトウエアの最新バージョンを各コンピュータにダウンロード及び配布する。ネットワーク・アドミニストレータはこのインベントリ及び更新プロセスを中心的に制御する。ＣＤ－ＲＯＭの配布は前述の費用の問題を有する。

【００１６】

【発明が解決しようとする課題】本発明の第１の局面によれば、コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入された１つ又は複数のコンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントが提供される。そのアップデータ・コンポーネントは、１つ又は複数の必要なソフトウエア・リソースが置かれているそのネットワーク内の１つ又は複数のロケーションを識別するための情報、前記１つ又は複数の必要なソフトウエア・リソースを検索するために前記１つ又は複数のロケーションへのアクセスを開始するための手段、及び前記１つ又は複数の検索されたソフトウエア・リソースを使用して前記導入されたコンピュータ・プログラムの１つにソフトウエア更新を施すための手段を含む。

【００１７】本発明によるアップデータ・コンポーネントは、関連するソフトウエア製品のアップグレード及びその製品におけるバグの修復を、更新基準に関する初期の同意後、ユーザによる如何なる対話も必要とすることなく自動的に制御することが望ましい。更新基準は、製品のライセンシング条件と関連付けることが可能である。これは、エラーがユーザの観点から自動的に訂正される場合、適切な更新ポリシを選んだユーザがいつも最も最新の利用可能なソフトウエアを持つことができることを保証する。ユーザは、アップデータ・コンポーネントがこれを処理するので、ソフトウエア更新が生じる場所、それらを得る方法、又はそれらを導入する方法を知る必要がない。ソフトウエア・ベンダは、エラーを訂正するために特別のＣＤ又はディスケットを出荷する必要がなくなるし、更なる特徴を与える必要もなくなる。ベンダは、顧客が新しい製品の特徴を早く及び努力することなく受けるように、漸次的増加を基準にして容易にコ

ードをリリースすることができる。

【００１８】本発明の望ましい実施例によるアップデータ・コンポーネントはローカル・コンピュータ・システムにおける利用可能なソフトウエア更新と導入済みソフトウエアとの間の比較を行い、どれが導入されたソフトウエアと関連しているかを識別し、利用可能な関連の更新とローカル・コンピュータ・システム上に保持された更新基準（これらの更新基準は現在のシステム又はシステム・ユーザに対して事前定義される）とを比較し、しかる後、その事前定義された基準を満足するソフトウエア更新を自動的にダウンロードし、その更新を施す。

【００１９】このように、ソフトウエア更新を自動的に施すことは、事前定義された更新基準と、更新を必要とするプログラム・コードと共にダウンロードされる導入のための命令との両方に従って、利用可能なソフトウエア・パッチ及び／又はアップグレードされたバージョンを導入することを伴う。ダウンロードされた命令を動的に実行するという特徴は、アップデータ・コンポーネントにより処理可能な更新のタイプに関連して融通性を与える。それは、単一の汎用アップデータ・コンポーネントを多くの種々のソフトウエア製品と共に使用することを可能にするためにも使用可能である。別の方法として、或るソフトウエア更新のための導入命令を、アップデータ・コンポーネント内で事前符号化してもよい。その「ソフトウエア・リソース」は、一般に、プログラム・コード、機械読み取り可能な導入命令、及びアドレス情報のような任意の必要なデータ変更の組合せである。

【００２０】ネットワーク・ロケーションを識別する場合に使用するための情報は明示的ネットワーク・ロケーション情報であってよく、或いは、それはソフトウエア・ベンダ名又はそのロケーションを識別するためのサーチ・パラメータとして使用可能な他の任意の情報であってもよい。望ましい実施例では、その情報は、その製品に更新を施すためのソフトウエア・リソースを記憶された関連のネットワーク・ロケーションを識別するためのサーチを開始するために、アップデータ・コンポーネントによってサーチ・エンジンに供給される製品識別子である。このサーチは、アップデータ・コンポーネントによって呼び出される通常のインターネット（又は他のネットワーク）サーチ・エンジンによって遂行可能である。サーチ・エンジンがネットワーク・ロケーションの識別情報を戻す時、アップデータ・コンポーネントは利用可能な関連の更新のリストをこのロケーションから検索し、ローカル的に保持されたソフトウエア製品バージョン及び事前定義された更新基準に関してそのリストをチェックし、それらの基準が満たされる場合、更新リソースを検索してローカル・コンピュータ・システム上に与える。

【００２１】本発明の望ましい実施例によれば、ソフトウエア更新を形成すべきソフトウエア・リソースに対して、標準化されたネーミングの常套手段が使用され、しかもアップデータ・コンポーネントは、ネットワーク・オペレーティング・システムのファイル・システムにおいてこれらのリソースに関してサーチすることができる。これは、ネットワーク可用性問題を緩和するためにソフトウエア・リソースが複数のロケーションに記憶されることを可能にし、開発者及び配布者がそれらのエラー修復パッチ及びソフトウエア製品のアップグレードされたバージョンを提供することを容易にする。例えば、開発者は、それらのＬＡＮにおける既知のファイル名を使用した公衆ネットワーク・ディスク・ドライブを介して、又は既知のキー・ワードを使用するためにサーチ可能な公開されたユニフォーム・リソース・ロケータ（ＵＲＬ）を介して得られる新しいソフトウエア更新を作ることができる。

【００２２】アップデータ・コンポーネントは、それらが更新するために働く製品の不可欠な部分であることが望ましい。従って、アップデータ・コンポーネントは、ソフトウエア製品の初期バージョンと共にソフトウエア・ユーザに配布される。しかる後、アップデータ・コンポーネントは、プリセット基準（更新に関する相次ぐサーチの相互間の期間のような、及び特定のユーザがすべての更新を受けることを選択したか、或いは、例えば、パッチの更新を受けるが置換製品バージョンを受けないというような或る更新だけを受けることを選択したか）に従ってソフトウエア更新を自動的に得て、それを施す。

【００２３】アップデータ・コンポーネントの更新機能は更新そのものを含むことが望ましい。実に、アップデータ・コンポーネントが、それの関連のソフトウエア製品を更新するためのソフトウエア・リソースを求めてそれがサーチする前に、それ自身に対する更新を得るために適正なネットワーク・ロケーションをいつもアクセスするように、更新基準はセット可能である。

【００２４】本発明によるアップデータ・コンポーネントは、前提製品が利用可能であるかどうかをチェックするための手段を含むことが望ましく、しかも、現在の製品に対する更新パスを選択するプロセスの部分として、必要なバージョンに同期化されることが望ましい。望ましい実施例では、それらの可用性をチェックすること以外に、アップデータ・コンポーネントは、これが同意済みの更新ポリシである場合、前提ソフトウエア製品と関連したアップデータ・コンポーネントにそれらのソフトウエアに対する更新を開始するように命令することができる。各ソフトウエア製品のアップデータ・コンポーネントが前提製品に対する更新をトリガすることができる場合、更新は、ユーザがその更新に関与することなく又はその更新を知ることを必要とすることなく、導入されたソフトウエア製品のセットを通して波及することができる。この機能は、更新が行われる時に非同期ソフトウ

エア・バージョンの問題を扱わない従来のアップデータ・エージェントに比べて大きな利点であり、エンド・ユーザのためのタスクを遂行するために配布オブジェクト相互間のコラボレーションに関してソフトウエア業界における増加する動向を支援する。

【００２５】アップデータ・コンポーネントは、暗号アルゴリズムを使用して、ダウンロードされたソフトウエアの真偽を検査するための機構を含むことも望ましいことである。これは、専用の、パスワード保護された、及び別の方法で保護されたソフトウエア・リソース・リポジトリ・サイトを必要ないものにする。ソフトウエア・リソースは、それらが正しく名前を付けられ、ネットワーク・サーチ・エンジンに通知される限り、ネットワーク上のどこにあってもよい。

【００２６】従って、本発明は、ソフトウエア更新を得るための及びその更新を施すためのエージェント及び方法を提供する。この方法は、ソフトウエア更新を配布及び追跡するソフトウエア・ディストリビュータに関するコスト及び労力を大幅に減らし、導入されたソフトウエアに更新を施すシステム・アドミニストレータ及びエンド・ユーザに関する労力を大幅に減らす。

【００２７】

【課題を解決するための手段】図１に示されるように、アップデータ・コンポーネント２０が、通常のネットワーク接続されたコンピュータ・システム１０のシステム・メモリに、関連のコンピュータ・プログラム３０と共に導入される。そのアップデータ・コンポーネントは、ローカル・コンピュータ・システムのユーザが導入するために、記憶媒体（ディスケット又はＣＤ）によってユーザに配布されてもよく、或いは他のコンピュータ・システムから明示的にダウンロードされてもよい。本発明の望ましい実施例では、アップデータ・コンポーネントは、それらが保守することを意図された（又は、それとは別に、同じ機構を介して及びそれらが関連したプログラムと同時に配布される）コンピュータ・プログラム内に統合される。アップデータ・コンポーネントは、ユーザがそれを取得し又は活性化するために如何なる特別のアクションを取ることも必要ないように、その関連のプログラムの導入手順の一部として導入される。各アップデータ・コンポーネントの導入は、そのアップデータ・コンポーネントがオペレーティング・システムと共にそれ自身を登録することを含み（更に一般的には、アップデータ・コンポーネントは、中心的な又は分散したリポジトリ４０を登録し）、従って、少なくともローカル・システムにおけるアップデータ・コンポーネントは、レジスタ・エントリにおけるアドレス情報及び／又はそれらの製品識別子によって識別可能であり、制御可能である。

【００２８】本発明の望ましい実施例の特徴は、各アップデータ・コンポーネントが、他のソフトウエア製品を管理する他のアップデータ・コンポーネントを見つけることができ、他のアップデータ・コンポーネントによって見つけられ、しかも他のアップデータ・コンポーネントとコミュニケートすることができるということである。この機能は、一方のアップデータ・コンポーネントが他方のアップデータ・コンポーネントを、前者が後述のようにそれ自身の更新を実行する前に特定のレベルに更新するために必要とする時使用される。これは、アップデータ・コンポーネントがオペレーティング・システム又は他のリポジトリ４０内に登録することによって可能にされる。

【００２９】望ましい実施例では、各登録エントリは２つの項目、即ち、アップデータ・パス及びアップデータ・ネットワーク・アドレスを含む。そのパスは、アップデータ・コンポーネントが、ブート・アップ・プロセス中、オペレーティング・システムによって立ち上げられるように、アップデータ・コンポーネント・バイナリ・ファイルのロケーションである。これは、アップデータ・コンポーネントがいつもアクティブであり、作業を遂行するか又は他のアップデータ・コンポーネントからそれに発生されたリクエストを処理する準備ができていることを保証する。ネットワーク・アドレスは、ネットワークにおける他のコンピュータ・システム上のコンポーネントによって、ネットワーク上のそれを見つけるために及びそれとコミュニケートするために使用されるアドレスである。

【００３０】ＵＮＩＸ（商標）オペレーティング・システム及びＴＣＰ／ＩＰプロトコル・スイートを使用するそのような登録の例は、アップデータ・コンポーネントに対する次のようなネーミング規約

SoftwareVendorName+_product_name+_updater

を使用する。

【００３１】パス登録は、パス・エントリを記憶するためにＵＮＩＸ/ｅｔｃ/ｉｎｉｔｔａｂファイルに入れられる。例えば、ＩＢＭ社のＤＢ２（商標）データベース製品に対するアップデータ・コンポーネントが導入される時、それは次のようなフォーム、即ち、

ibm_db2pe_updater:2:respawn:/usr/abin/db2_updater_binary

という /ｅｔｃ/ｉｎｉｔｔａｂファイルにエントリを加えるであろう。

【００３２】コンピュータ・システムがリブートする度に、それはこのファイルを読み取り、ＤＢ２アップデータ・コンポーネントを立ち上げるであろう。アップデータ・エントリにおける「ｒｅｓｐａｗｎ」キーワードは、アップデータ・コンポーネント・プロセスが一般的なシステム・オペレーション中に何らかの理由で障害を起こした場合、それはオペレーティング・システムによって自動的に再始動されることを保証する。この方法は、すべての導入されたアプリケーションに対するすべ

てのアップデータ・コンポーネントがいつもアクティブであることを保証するであろう。

【００３３】ネットワーク・ロケーション登録は、ＵＮＩＸ/ｅｔｃ/ｓｅｒｖｉｃｅファイルに入れられる。例えば、ＤＢ２アップデータ・コンポーネントが導入される時、それは次のようなフォーム、即ち、

ibm_db2pe_updater　5000/tcp　#net location of DB2 updater component

という /ｅｔｃ/ｓｅｒｖｉｃｅｓ ファイルにエントリを加えるであろう。

【００３４】別のアップデータ・コンポーネントがＤＢ２アップデータ・コンポーネントとコミュニケートすることを望む時、それは、ＤＢ２アップデータ・コンポーネント名 ibm_db2pe_updater に関してこのファイルをサーチすることによってそれを見つけるであろう（実際には、ＵＮＩＸコール getservbyname() によって間接的に行われる。その名前は標準的なネーミング規約に従って発呼者によって形成される）。それが見つかる時、それはＤＢ２アップデータがポート番号５０００における接続に関して聴取していることを知り、ＴＣＰプロトコルを使用するであろう。これは、問題のアップデータ・コンポーネントがＤＢ２アップデータ・コンポーネントへのリンクを確立して会話（これについては後述する）を開始することを可能にする。

【００３５】アップデータ・コンポーネントが他の遠隔の機械における他のアップデータ・コンポーネントを見つけてそれと通話するためには、上記情報は、両方の機械からアクセス可能であり且つそれを必要とするすべてのアップデータ・コンポーネントにとって利用可能なリポジトリ４０'（望ましくは、ウェブ・ページ又はパン-ネットワーク・ファイルのようなそのネットワークにおける何処からもアクセス可能な中央データベース又は分散データベース）を持つことによって拡張されなければならないであろう。エントリは、updater_name machine_ip_address（又は、ＤＮＡエントリ）、ポート番号、プロトコルという形式のものであろう。

【００３６】例えば、或る機構の製造部門は、分散ソフトウエア製品が相互に協同する３つのコンピュータ・システムを有することがある。それらのシステムはａ、ｂ、及びｃと呼ばれる。ウェブ・ページ又はファイル manufacturing_collaborators.html における代表的なエントリは次のようになるであろう。

ibm_catia_updater　a.manufacturing.com　5000 tcp

ibm_db2pe_updater　b.manufacturing.com　5100 tcp

ibm_cics_updater　c.manufacturing.com　4780 tcp

【００３７】次に、アップデータ・コンポーネントは、ＩＰアドレスを作成するためのＤＮＡ名と、遠隔のアップデータ・コンポーネントがそのアドレスにおいて聴取しようとしているポート番号とを使用して他の任意のアップデータ・コンポーネントを接続し、それと通話することができる。

【００３８】従って、導入時のアップデータ登録のステップは次のようになる。

（１）/ｅｔｃ/ｉｎｉｔｔａｂ　ファイルにおけるエントリを作成する（アップデータ・プロセス・コード・ロケーションを登録する）

（２）/ｅｔｃ/ｓｅｒｖｉｃｅ　ファイルにおけるエントリを作成する（アップデータ・プロセス・ローカル・アドレスを登録する）

（３）中央データベース・ファイルにおけるエントリを作成する（アップデータ・プロセス・パン-ネットワーク・アドレスを登録する）

【００３９】導入プロセスはアップデータ・コンポーネントにウェブ・プロキシ・サーバのローカルＩＰアドレスを供給することも伴うことがある。多くの代替の登録実施方法が可能であることは当業者には明らかであろう。

【００４０】アップデータ・コンポーネントは、それらが関連するソフトウエア製品に対する識別子及びバージョン番号のためのデータ・フィールドを含む。アップデータ・コンポーネントは、ヌル値にセットされたこれらのフィールドと共に顧客に配布可能であり、従って、導入手順は、アップデータ・コンポーネントが、識別子、現在のプログラム・バージョン、及びリリース番号を得るためにそれのソフトウエア製品に質問するという初期ステップを含む。別の方法として、ソフトウエア・ベンダは、アップデータ・コンポーネント内に関連の製品ＩＤ及びバージョン番号を事前コード化することも可能である。

【００４１】図１のシステム１０は、複数の遠隔サーバ・システム５０、５０'を含むコンピュータのネットワーク１００において接続されたものが示される。ローカル・システム１０上に導入されたプログラムに更新を施すためのソフトウエア・リソースが遠隔サーバ・システム５０、５０'から得られる。各サーバ・システムは、そのサーバから得られるソフトウエア製品の最新バージョンのリスト６０及びそのソフトウエア製品に対するパッチを記憶装置内に含む。各ベンダは、アップデータ・コンポーネントによって読み取り可能なフォーマットで製品リリース・ヒストリを含むソフトウエア更新のリスト６０（その例が図２に示される）のようなウェブ・サイトを利用するものと仮定し、しかも所与のレベルから新しいレベルへのリリース（ソフトウエア製品リリースから新しいレベルへのこの遷移は以下では「成長パス」と呼ばれる）を形成するために必要なソフトウエア・リソース７０を利用するものと仮定する。ソフトウエア更新リスト6.0におけるエントリは、各ソフトウエア製品

バージョン１１０に対して、その更新を施すために必要なソフトウエア・リソースの識別情報１２０及びそれの前提ソフトウエア製品及びそれらのバージョン番号の識別情報１３０を含む。或る場合には、必要なリソースはソフトウエア及び関連の導入命令の完全な置換バージョンである。別の場合では、リソースは、（例えば、エラー訂正のために）既存のプログラムを修正するためのパッチ・コード及びパッチの導入命令を含む。

【００４２】現在の例に関して、ネットワーク１００はインターネットであると仮定することにするが、本発明は任意のコンピュータ・ネットワークにおいて実施可能である。ネットワーク１００にはサーバ・システム８０が示され、そのサーバ・システムには、ネットワーク上の更新ソース・ロケーションを見つける場合に使用するためのサーチ・エンジン９０が導入される。これは、ローカル・システム１０から離れて設置されるように示されているが、必ずしもその必要はない。図面では、各アップデータ・コンポーネント２０は単一のプログラム３０と関連付けて示され、しかも、すべての導入されたソフトウエア製品がそれらを管理する関連のアップデータ・コンポーネントを有することは本発明のこの実施例の１つの特徴であるが、後述するように、これらの特徴のうちのどれも本発明にとって本質的なことではない。

【００４３】次に、図３、図４及び図５を参照して、アップデータ・コンポーネントのオペレーションを説明することにする。導入されたアップデータ・コンポーネントが実行される時、それの第１アクションは、導入時に得られた製品識別子及び製品バージョン・リリース番号をサーチ引数として１つ又は複数のサーチ・エンジン９０に供給し、特定のソフトウエア製品に対して利用可能な更新に関するサーチを開始することである（ステップ２００）。ソフトウエア・ベンダは、製品識別子及びリリース番号１１０によって参照される利用可能な製品更新のリスト６０をそれらベンダのウェブ・サイトを介して供給するものと仮定すると、そのサーチは、更新情報が得られる関連のウェブ・サイト１４０を識別するであろう。サーチ・エンジンを始動しようとする初期の試みが不成功である場合、アップデータ・コンポーネントは異なるサーチ・エンジン（第１のサーチ・エンジンに対して異なる地理的ロケーションにあってもよい）を始動するように試みるであろうが、別の方法として、事前設定された期間を待ち、しかる後、再試行してもよい。更新情報に対する関連のウェブ・サイト１４０を識別するＵＲＬはサーチの結果としてアップデータ・コンポーネントに戻される（ステップ２１０）。

【００４４】アップデータ・コンポーネントはそのＵＲＬを使用してリスト６０をアクセスし（ステップ２２０）、特定の製品に関連する利用可能な更新のリスト６０の部分を含むファイル１６０をダウンロードする（ステップ２３０）。しかる後、アップデータ・コンポーネントは、図４及び図５に示されるステップ２４０－２８０を遂行する。各ファイル１６０は、ディジタル的にサインされたメッセージ・ダイジェスト（例えば、ＭＤ５）を含む。次に、検索されたファイル１６０が、ディジタル・シグニチャ・チェック・アルゴリズム（米国特許第５，２３１，６６８号に開示されたアルゴリズムのような）を使用して分析される（ステップ２４０）。これは、ファイル１６０が特定のソフトウエア製品に対する正しいソフトウエア更新リストを表すこと、及びそのファイルがサイン以後修正されていないことを確認するためには重要である。更に、ディジタル・シグニチャのチェックは、これらが正しいもの以外に複数のウェブ・ページＵＲＬを含み得るので、サーチの結果を濾す有用な方法である（サーチは、ソフトウエア・ベンダによって発行されなかったページを含む、名前を付けられた製品バージョンに対する参照符合を持った他のページを見つけることがある）。ファイルをダウンロードし、それを検証しようとする試みが成功しなかった場合、アップデータ・コンポーネントはそのサーチにおいて見つかった次のＵＲＬに進む。

【００４５】次に、アップデータ・コンポーネントは、ローカル・コンピュータ・システムにおいて、現在導入されているソフトウエア製品の識別子及びリリース番号と検索されたファイル１６０におけるリストされた利用可能な更新との間で比較を行う（ステップ２５０）。この比較は、現在のバージョンから更新バージョンへの可能な成長パスを決定するが、これらの可能な成長パスは、その後、事前定義された更新基準と比較され（ステップ２６０）、更新基準を満たさない如何なる可能な成長パスも廃棄される。従って、アップデータ・コンポーネントは、現在のソフトウエア製品から利用可能な新しいバージョンに移行することが可能であるかどうか、及び現在同意されているライセンス条件の下でパッチを現在のバージョンに適用することが可能であるかどうかを決定する。

【００４６】例えば、ソフトウエア製品のライセンスは、その製品に関する将来の如何なるバージョンへの移行及び利用可能な如何なるパッチの適用を可能にし、又は指定されたバージョンへの移行だけを可能にすることがあり、或いは、現在のバージョンにおけるエラーを修正又は訂正する利用可能なパッチの適用を許容するだけであることもある。現在のライセンスの制限のために利用し得ない可能な更新パスは、現在導入されているバージョンのソフトウエア・アセット・マネージャ（エンド・ユーザ又はＩＴ調達マネージャでもよい）に送られたシステム発生のメッセージとして通知され（ステップ２７０）、現在のライセンスが十分なものであるかどうかに関してそれらが判断することを可能にする。

【００４７】起こり得る更新に関するライセンス制限の他に、アップデータ・コンポーネントの更新基準又は成

長ポリシは、サイクル期間（例えば、１週間又は１ヶ月間）と、複数の可能な成長パスのうちのどれを選択すべきか（ライセンスによって許容される最新のバージョンをいつも選択するようにするか、又は最新のパッチをいつも選択し且つ新しいバージョンの利用可能性を通知するだけであるようにするか、或いは、前提ソフトウエアが既にローカル・システムにおいて得られる場合には新しいバージョンを選択するだけであるようにするか）を決定するための基準とを含む。成長基準は、アップデータ・コンポーネントによってダウンロードされる新しいバージョンにいつアップグレードすべきかというような制御情報も含むことがある。新しいソフトウエア製品バージョンに移行する時にデータ・マイグレーションが要求される場合、これが業務時間外で、或いは毎月又は毎年単一のスケジュールされた時間においてのみ行われることが必須であることがあり、これはアップデータ・コンポーネントによって制御可能である。

【００４８】成長ポリシの定義は、現製品との同期を維持する必要がある時には前提ソフトウエア製品の更新がリクエストされるべきであるということを決定するパラメータを含むこともある。これについては、更に詳しく後述することにする。アップデータ・コンポーネントによって設定及び適用される基準には大きな融通性があることは当業者には明らかであろう。

【００４９】次に、アップデータ・コンポーネントは、更新基準を使用して可能な成長パスのセットから特定の成長パス（即ち、アップグレードするために利用可能なバージョン）を決定する（ステップ２８０）。例えば、アップデータ・コンポーネントは、更新基準によって許容される利用可能な更新のうちの最高の可能なバージョン又はリリース番号を、それが更新ポリシである場合、選択してもよい。

【００５０】アップデータ・コンポーネントは、必要なソフトウエア・リソースが既にローカル・コンピュータ・システムにおいて利用可能であるかどうかをチェックするためにオペレーティング・システムのファイル・システムのスキャンを遂行する（ステップ２９０）（図３、図４及び図５参照）。その必要なリソースは、現在のアプリケーション・ソフトウエアを新しいレベルにもたらすために必要なソフトウエア更新加工物であり、前提ソフトウエアを必要なレベルに更新することを必要とするソフトウエア更新である。前提導入済み製品と関連した各アップデータ・コンポーネントは、（ａ）それが導入されること、及び（ｂ）それが、必要な前提レベルにあるか或いはそれよりも高いレベルにあること、を保証するために接触される（ステップ３００）。すべての必要なリソースがローカルで又は他の機械において得られ且つ確認された場合、アップデータ・コンポーネントは、更新されたソフトウエア・バージョンを形成するステップ３１０（図５参照）に進む。それが確認されない場合、アップデータ・コンポーネントは必要なリソースを得なければならない。

【００５１】図３、図４及び図５に示されるように、更新されたバージョンを形成するための必要なソフトウエア・リソースがローカル・システムにおいて見つからなかった場合、アップデータ・コンポーネントは、その必要なリソースを見つけるために１つ又は複数のサーチ・エンジンに更なるリクエストを供給する（ステップ３２０）。サーチ・エンジンは１つ又は複数のＵＲＬを返送し（ステップ３３０）、アップデータ・コンポーネントはこれらを使用してソフトウエア・リソースを検索してローカル・コンピュータ・システムの記憶装置に入れる（ステップ３４０、３５０）。この段階では、アップデータ・コンポーネント又はユーザは、新しいバージョンにどのような訂正又は機能強化が含まれるかということに関する知識を全く持つ必要がない。ユーザがすべての更新の内容を調べるという労力を免れるように、更新基準は、どのタイプの更新が必要であるかを決定する。実際には、ユーザが更新の影響を決定することができることが望ましく、従って、更新のためのソフトウエア・リソースは、ユーザ又はアドミニストレータが読むことができるこれらの影響に関する記述を含む。

【００５２】例えば、更新されるべきソフトウエア製品はワード・プロセッサ・アプリケーション・プログラムであってもよい。販売されたワード・プロセッサが或るフォントを脱落していたり或いはシソーラスを含まなかったりした場合、その後、これらのフィーチャを加えるためのパッチが利用可能にされることがある。アップデータ・コンポーネントは、更新基準次第でこれらをそのワード・プロセッサに加える機能を有する。

【００５３】本発明の別の実施例では、更新リストに関する初期サーチに続く必要なソフトウエア・リソースに関するサーチは不必要である（或いは、前提ソフトウエア製品及びパッチ又は現製品に対する新しいバージョンがある場合に必要であるだけである。下記参照）。これは、現製品によって直接に必要とされる更新ソフトウエア・リソースが必要なリソースのリストと関連して記憶されるためである。即ち、そのリストは、そのリストからの成長パスの選択が必要な更新のネットワーク・ロケーションに対するポインタ（場合によっては、更に、前提ソフトウエア製品のロケーションに対するポインタ）の選択を伴うように、その必要なリソースのネットワーク・ロケーションに対するポインタを含む。

【００５４】ディジタル・シグニチャ・チェックによる第２の検証が、今度は、ダウンロードされたリソースに関して遂行される（ステップ３６０）（図５参照）。ダウンロードされたリソースの正当性を検証した後（ステップ３６０）、アップデータ・コンポーネントは、更新ポリシに従ってターゲット環境における導入を自動的に行う（ステップ３１０）。実際には、これは、管理パス

ワードのようなユーザからの情報又はデータベース使用パラメータ値を必要とすることがあるが、本発明の望ましい実施例では、ダウンロードされたコードの導入は、ユーザが何らかの導入情報を余所から知る又は余所から得ることをそれが必要としないという意味で、及び、アップデータ・コンポーネントが自動的に更新を施すことを事前定義の更新基準が可能にする場合、一般に、ユーザが実行時に如何なる判断も行わなくてよいことをそれが可能にするという意味で自動的である。

【００５５】シェルにおいてエンコードされた機械読み取り可能な導入命令を（例えば、Ｓｃｒｉｐｔ、又はＰＥＲＬのような解釈言語、又はマイクロソフト社のＷｉｎｄｏｗｓ（商標）オペレーティング・システム上のアプリケーションの場合の setup.exe のような実行可能なものとして）含むことはよく知られている。本発明によるアップデータ・コンポーネントは、機械読み取り可能な命令を関連のソフトウエア・リソースと共にダウンロードし（ステップ３５０）、それらを自動的に実行するであろう（ステップ３１０）。従って、アップデータ・コンポーネントは導入命令を自動的に処理し、従来方法では必要とされた人からの入力を回避する。Ｓｃｒｉｐｔは、アップデータ・コンポーネントの第１バージョンを導入した第１の導入者から集められた情報（例えば、ユーザ名及びアプリケーション・アドミニストレータのパスワードのような情報、及び導入ディレクトリ等）を再使用するように適応可能である。

【００５６】本発明の望ましい実施例による更新の方法は、ソフトウエア・ベンダが、形成する必要のあるソフトウエア・リソースを１つの製品レベルから他の製品レベルに編成することを必要とする。例えば、バージョン１．１．１からバージョン１．１．４への移動は、一般には、施されるべき一連のパッチ、及び機械処理可能な導入命令にうまくエンコード可能である場合の必要な導入順序を含むであろう。そこで、ユーザは、修復及び機能強化を施す順序をユーザが制御することを必要とする方法では固有である努力及び人的エラーの危険を免れる。従って、１つの製品レベルから別の製品レベルに如何に移行すべきかという問題は、顧客に代わってソフトウエア・ベンダによって処理され、アップデータ・コンポーネントは、ベンダによってサポートされるレベル（即ち、特定の既存の製品レベルに対してソフトウエア・ベンダにより発行された成長パス）までしか移ることができない。

【００５７】アップデータ・コンポーネントはレポートを発生し（ステップ３８０）、ログ・レポートに書き込み（ステップ３９０）、しかる後、所定の更新サイクル期間（反復期間パラメータは、アップデータ・コンポーネント導入された時に形成される）の満了時に再び活性化される（ステップ４１０）まで実行を終了する（ステップ４００）（望ましい実施例では、アップデータがスリープ又はアイドル状態に進む）。

【００５８】Ａ１．アップデータ・コンポーネントの構造

アップデータ・コンポーネントの構造は、データ、そのデータを操作するための方法、及び他のアップデータ・コンポーネントがそれと接触及びコミュニケートすることを可能にする汎用アプリケーション・プログラミング・インターフェースより成る。次に、この構造を詳しく説明することにする。

【００５９】アップデータ・コンポーネント・データ：アップデータ・コンポーネントはつぎのような永続的データを含む：

Product_ID：　このアップデータ・コンポーネントによって管理されるソフトウエア製品の識別子。

Current_Installed_Version：　導入されたソフトウエアに対するバージョン識別子（例えば、バージョン３．１．０）。

Current_License：　現在のソフトウエア・ライセンスによってユーザがアップグレードし得るソフトウエア製品バージョンに対応するバージョン識別子（例えば、４．０．ｚ）。別の方法として、これは、機械読み取り可能なライセンス条件をアクセスする時に使用するためのライセンス識別子（例えば、ＬＩＣＩ）であってもよい。

Installation_Environment：　属性名／属性値の対のリスト。これは、アップデータ・コンポーネントが初めて使用された時、ユーザによって入れられた値を記憶するためにアップデータ・コンポーネントによって使用される。例えば、アップデータ導入ユーザＩＤ及びパスワード、或いは、ルート・パスワード、導入ディレクトリ、ウェブ・プロキシ・サーバ・アドレス、サーチ・エンジンＵＲＬ、ログ・ファイル名、ソフトウエア・アセット・マネージャ電子メール・アドレス等。このデータは、その後の自動更新が必要とされる時に再使用されるであろう。

成長ポリシ・パラメータ：

ａ．Growth_Cycle：　アップデータ・コンポーネントが毎日、毎週、又は毎月それのソフトウエア製品を更新しようと試みなければならないかどうかを決定するデータ。

ｂ．Growth_Type：　更新がバグ修復及び機能強化（即ち、パッチ）のみに制限されるか、或いは、各成長サイクルにおける最新のリリースへのアップグレードを必要とするかを決定するデータ。

ｃ．Force_Growth：　（イエス／ノー）アップグレードするように他のソフトウエア・リソースを強制すべきかどうかを、それがこのソフトウエアのアップグレードのための前提である場合に決定するパラメータ。（或る実施方法は、アップグレードするように他のソフトウエアを強制することによって、この単純なイエス／ノーより

ももっと融通性のある制御を提供するであろう）。
Last_Growth_Time：　アップデータ・コンポーネントが最後に実行された時の日付及び時間。
【００６０】アップデータ・コンポーネントは次のような非永続的データも含む：
Possible_Growth_Paths：　利用可能なアップグレード・パスを表す一時データ（例えば、バージョン番号３．１．ｄ、３．２．ｅ、４．０．ａ）。
【００６１】Ａ２．私的アップデータ機能：アップデータ・コンポーネント・ロジックは次のような方法を含む：
Discover_Possible_Growth_Paths()：
インターネット（又は、他のネットワーク）におけるこのソフトウエアのための Growth_Path 情報に関するサーチ。このサーチ方法は、標準的なサーチ・エンジン・サーバを介してサーチを開始する。戻された情報は更に新しいバージョン及び関連の前提製品情報である。次に、Growth_Path 情報は成長ポリシパラメータに従って減少する。Growth_Path リストにおけるすべてのメンバに対して、前提製品の適正なバージョンがローカル・コンピュータ及び／又はリモート・コンピュータにおいて得られるかどうかに関するチェックが行われる。これらの前提製品を管理するアップデータ・コンポーネントはアクセスされ、これがポリシである場合、成長することを強制される。すべての事前製品が正しいレベルでローカルに存在する場合、又はネットワーク上で遠く離れて得られ、しかも、「強制成長」ポリシと共に存在する場合、ソフトウエア製品のより新しいバージョンに対するｉ識別子が Possible_Growth_Paths リストに加えられる。
Decide_Growth_Path()：
成長ポリシを解釈し、単一の成長パスを選択する。本発明の或る実施方法は、例えば、他のプログラムへの更新を強制すべきかどうかというような考察事項が存在する場合、ユーザ対話がそのパスを選択することを伴うであろう。
Get_Resources(Parameter: Chosen_Growth_Path)：
Chosen_Growth_Path（例えば、３．２．０）を与えられた場合、必要なリソース（パラメータ Product_ID、Current_Installed_Version、Chosen_Growth_Path）に関してサーチし、すべてのリソースをローカル・コンピュータにダウンロードする。これは、新しいバージョン及び機械処理可能な導入命令を必要とするソフトウエアを含むであろう。
Install_Resources()：
必要なファイルを正しいロケーションに導入すること、或いは、それらのファイルをコンパイルすること、及びソフトウエアを収容するために既存のシステムの構成を修正することを含み、すべてのアクションをファイルにログする（及び「アンインストール」方法がすべてのアクションをやり直すことを可能にする）導入命令を処理する。
Grow()：
方法を開始する：
Discover_Possible_Growth_Paths()
可能な成長パスが存在しない場合、アップデータ・コンポーネントはアイドル状態になるさもなければ、
Decide_Growth_Path()
Get_Resources(Parameter: Chosen_Growth_Path)
Install_Resources().
Grow()は、すべての完了したアクションをログに書き込み、アップデータ・コンポーネントの実行を終了する。アップデータ・コンポーネントは、新しい更新要件に関して再びチェックすべき時間まで、又はそうするように他のアップデータ・コンポーネントによって指示されるまでアイドル状態になる。
【００６２】Ａ３．汎用アップデータ・コンポーネントＡＰＩ
アップデータ・コンポーネントは下記のような汎用ＡＰＩを含む。これらの機能は、リモート・プロシージャ・コール、メッセージ指向ミドルウエア，ＯＲＢ（オブジェクト・リクエスト・ブローカ）等のような既存のネットワーク通信ソフトウエアを使用して呼び出し可能であろう。
Get_Release()：
この機能は他のアップデータ・コンポーネントによって呼び出され、そしてこのアップデータ・コンポーネントによって管理される製品のリリース・レベルを戻す。
Update(new_level)：
他のアップデータ・コンポーネントがこの機能を呼び出し、このアップデータ・コンポーネントによって管理される製品を、new_level パラメータ値によって表された新しいレベルに移す。これは私的機能 Grow() を呼び出す。
Receive_Event(event details)：
アップデータ・コンポーネントがアップグレードするようにリクエストを受ける時、それは、それがいつ更新を完了したかを呼出のアップデータ・コンポーネントに知らせなければならない。他のアップデータ・コンポーネントに代わって更新を遂行するアップデータ・コンポーネントは、その更新の成功を伝えるためにリクエストしたアップデータ・コンポーネントのこの機能を呼び出すであろう。イベント詳細は「製品ＩＤ、新リリース・レベル、ｏｋ」又は「製品ＩＤ、新リリース・レベル、失敗」のようなストリングであってもよい。
【００６３】更新の強制を可能にすること（又は、更新の強制が更新ポリシの一部でない場合、ソフトウエア・アセット・マネージャに通知を送ること）による非同期化された前提製品の潜在的問題点の自動処理は、従来技術の更新方法よりも優れている重要な利点である。

【００６４】初期サーチに応答してアップデータ・コンポーネントに戻された更新リスト・ファイル１６０は前提ソフトウエアの識別子１３０を含むので、その情報は、前提ソフトウエアがローカルで入手可能に又はリモートで入手可能かを、アップデータ・コンポーネント登録データベース４０、４０'の前述の検査（ステップ２９０）がチェックすることを可能にする。それがローカルで設置された又はリモートで設置されたアップデータ・コンポーネントをすべて見つける場合、前提ソフトウエアが得られることは確かであり、次に、すべての前提が正しいレベルにあることを確実にするために各ソフトウエア製品に対する各アップデータ・コンポーネントと接触することが必要である。前提ソフトウエア３０'に対する必要な製品識別子を有するが、必要なバージョン番号を持たないアップデータ・コンポーネント２０'がローカルで又はリモートで見つかった場合、及び更新の強制が更新ポリシである場合、第１コンピュータ・プログラムのアップデータ・コンポーネント２０はこの前提アップデータ・コンポーネント２０'と接触し（ステップ３００）、それがそれの関連の前提ソフトウエア製品３０'の更新を試みることをリクエストする。このアップデータ・コンポーネント２０'は、必要な場合には、それの前提ソフトウエアの他のアップデータ・コンポーネントにそれらのバージョンを更新するようにリクエストする。

【００６５】或る段階において、関連のアップデータ・コンポーネントがローカルで又はリモートで見つからない場合、関連製品を更に成長させるためには、新しい製品に対する要件を知らせるためにアセット・マネージャに送られる。新しいレベルに成長させるためのアップデータ・リクエストの連鎖時の或る段階で、１つのアップデータ・コンポーネントが必要なレベルに移ることができなかった場合、この失敗はそれが呼び出したアップデータ・コンポーネントに報告され、そのアップデータ・コンポーネントはそのコンポーネント更新オペレーションの失敗等を、トランザクション全体を開始させたアップデータ・コンポーネントにプロンプト指示する。

【００６６】従って、それらの更新基準によって定義されたそれらの自動的な行為の他に、アップデータ・コンポーネントは他のアップデータ・コンポーネントからのリクエストのように外部刺激に反応することができる。

【００６７】Ｂ．更新同期化の例

次に、２つの製品の間の更新同期化の実施方法の例を説明することにする。この例は、すべての製品が存在し、互換性のあるリリース・レベルにあるように、一方のアップデータ・コンポーネントが前提ソフトウエアを同期化するように他方のアップデータ・コンポーネントとコミュニケートする方法を示す。

【００６８】ＣＯＲＢＡ（共通オブジェクト・リクエスト・ブローカ・アーキテクチャ）ＯＲＢ（オブジェクト・リクエスト・ブローカ）は２つのアップデータ・コンポーネントの間のロケーション及びコミュニケーションのために使用される。上記汎用ＡＰＩを使用すると、或るアップデータ・コンポーネントがネットワーク上の他のアップデータ・コンポーネントと会話することができるように、ＣＯＲＢＡプログラミングの技術に詳しい人がコミュニケーション・コードを開発することは簡単なことである。この例では、アップデータ・コンポーネント登録データベース４０は、各導入されたアップデータ・コンポーネントに対して「updater_component_name.iop」と呼ばれるファイルを含む、ネットワークを介して得られるディレクトリ又はフォルダである（なお、ｉｏｐは、インターオペラブル・オブジェクト・リファレンスを表す）。

【００６９】このファイルは、例えば、ＣＯＲＢＡ機能、即ち、

CORBA::Object::_string_to_object() in C++

を使用してファイルを読み取る任意のアップデータ・コンポーネントによってそのアップデータ・コンポーネントに対する基準に変換され得る一連のバイトを含む。

【００７０】更に、この基準は、対応するアップデータ・コンポーネントに対する独特のアドレスをそれが表すので、ネットワーク上のどこにあるアップデータ・コンポーネントに対しても基準になり得る。アップデータ・コンポーネントＡがアップデータ・コンポーネントＢに対する基準を作った時、アップデータ・コンポーネントＡは、例えば、C++ mapping A->Get_Release() を使用することによって汎用ＡＰＩ関数を呼び出すことができる。それは、その後、アップデータ・コンポーネントＡによって管理されるソフトウエアのリリース・レベルの値を戻すであろう。

【００７１】この例では、２つの製品、即ち、それぞれ異なる機械M及びNにおけるＩＢＭ社のＤＢ２製品及び「照会ビルダ」と呼ばれる照会ツールを考察することにする。（機械M及びNは同じ機械であってもよい；この例は単に、それらが別個であってもよいことを示す）。両方の製品とも、簡単に概説したように、ＣＯＲＢＡＯＲＢアーキテクチャを使用するアップデータ・コンポーネントを有する。ＯＲＢコミュニケーション・デーモンは参加システムM及びNにおいてアクティブである。

【００７２】ステップ１．　登録フェーズ：ＤＢ２アップデータ・コンポーネントは、オペレーティング・システムがシステムMにおいて始動し、ネットワーク・ファイル・システム・フォルダ又はディレクトリにおける ibm_db2_updater.iop と呼ばれるファイル（そのファイルに関するその後のサーチを助けるために使用される或るネーミング標準による）を直ちに作成する。このディレクトリは必ずしもM又はNではない任意の機械においてホストにされる。そのファイルは、アップデータ・コンポーネントに対する基準を作るために使用可能な一連

のバイトを含む。

【００７３】［擬似コード］

```
Filehandle=open("/network/filesystem/directory","ibm_db2_updater.iop");
ReferenceBytes=CORBA::Object::_object_to_string();
Write(FileHandle, ReferenceBytes);
close(Filehandle);
```

【００７４】QueryBuilder アップデータ・コンポーネントが始動してそれの登録を同じディレクトリ又はフォルダに書き込み、この場合には、再び、ファイル ibm_querybuilder.iop を呼び出す。

【００７５】この段階では、両方のアップデータ・コンポーネントがアクティブであり、それらの存在及びロケーションをネットワーク・ディレクトリに登録している。

【００７６】ステップ２．QueryBuilder はバージョン１からバージョン２に成長しようとするが、前提はＤＢ２バージョン２．１又はそれ以上である。下記のアクション・シーケンスが生じるであろう。QueryBuilder はＱＢとして表され、ＤＢ２はＤＢ２として表される。

【００７７】ＱＢ：ネットワーク・ディレクトリにおいてファイル ibm_db2_updater.iop（標準に従って作られたファイル名）に関してサーチする。それはファイルを見つけ、それを読み取り、それを使用可能な基準に変換する。

【００７８】

```
[擬似コード]
if (dbref =
CORBA::Object::_string_to_object(readfile(ibm_db2_updater.iop)))
    then SUCCESS: 我々はアップデータに接続された
    else
            FAIL: 前提ソフトウエアはコラボレート・システムのセットには存在しない。ソフトウエア・アセット・マネージャに電子メールを送り、状況を通知する。
            新バージョンに成長する試みを中止する。
    endif
```

【００７９】ステップ３．この段階で、我々はネットワーク化されたコンピュータのセットにおけるどこかにＤＢ２が存在することを知る。今や、我々は、それが正しいレベルにあるかどうかを知る必要がある。我々は、ＱＢアップデータ内から上記の汎用ＡＰＩ機能 Get_Release() を実行することによってこれは簡単に行う。従って、ＱＢアップデータは、それに関して何かを行うように、即ち、それがどのようなリリースであるかを知らせるようにＤＢ２アップデータにリクエストするクライアントである。

【００８０】［擬似コード］

```
db2_release = dbref->Get_Release();
```

【００８１】例えば、これは値「２．０」を戻す。

※【００８２】ステップ４

クライアント側：ＱＢアップデータ・コンポーネントは、これが十分ではないことを知っており、それはバージョン２．１を必要とする。それはそれの Force_Growth パラメータを調べる。そのパラメータにおいて、例えば、「イエス」は、前提ソフトウエアを、それがそれ自身の更新プロシージャを遂行する前に必要なレベルまで成長させなければならないことを意味する。従って、ＱＢアップデータは、新しいリリースまで成長するようにＤＢ２アップデータに通知し、しかる後、前提ソフトウエアが新しいリリースに成長するまで、又がそうすることに失敗するまで待つ。

※【００８３】

```
[擬似コード]
dbref->Update("2.1", QBref); // QBref はＱＢアップ
                // データに対する既製の基準である。ＤＢ
                // ２アップデータがそれ自身を更新する試
                // みを終了した時、それが直ぐに成功又は
                // 失敗というその結果を送ることができる
                // ように、ＤＢ２アップデータに送られる
EVENT= null
While (EVENT equals null)
      {何もしない;}
if (EVENT equals "SUCCESS")
      then  このプロセッサ・コンポーネント（即、Query
            Builder）によって管理されるソフトウエアを
```

成長させようとする
else
失敗をログに書き込む;
成長させようとしない;
スリープに進み、その後トライする;
endif.

【００８４】サーバ側：ＤＢ２アップデータ・コンポーネントは成長するというリクエストを受ける。それは成長するように試みる。

【００８５】それは結果を呼出クライアントに報告する＊10

[擬似コード]
```
DB2 attempts to grow.
if Growth Successful then
   QBRef->Receive_Event("SUCCESS"); // 機能 Receive_
         // Event の実施はＱＢアップデータ・コンポ
         // ーネントにおける EVENT と呼ばれる変数を
         // ＡＰＩコールにおいて送られたパラメータ
         // の値、即ち、ＩＦステートメントのこのセ
         // クション内にある場合には "SUCCESS" に単
         // にセットすることに注意して欲しい。
else
   QBREF->Receive_Event("FAILURE");
end if
```

＊（それは機能呼出において呼出元に対する基準を受けるので、呼出クライアントと接触する方法を知っている）。

【００８６】

【００８７】前述のように、事前定義された更新基準は、利用可能な更新セットのうちのどれが適用されるべきか、及びどれが無視されるべきかを決定することができる。更新基準は、ソフトウエア更新が利用可能であるとして識別されるがこの更新の適用が更新ポリシにはなく或いは不可能である時、エンド・ユーザ又はシステム・アドミニストレータに通知を送るためのアップデータ・コンポーネントへの命令を含むことができる。前に示された例の１つは、更新ポリシが前提ソフトウエア製品のアップグレード又はデータのマイグレーションを必要とすることがあり（例えば、ソフトウエア製品がデータベース製品である場合）、一方、それが何らかのエラー訂正パッチを導入することを意図したポリシであり得るので、その更新ポリシがソフトウエア製品の完全置換バージョンを導入することではないということである。一方の製品をアップグレードすることが他方の前提相補製※品のアップグレードを必要とする場合、更新の自動導入よりもむしろ通知が実施可能である。

【００８８】更新ポリシは、アップデータがユーザ又はアドミニストレータからの入力をリクエストする環境を定義することによって、更新プロセスの自動化の程度も決定することができる。

【００８９】次に、特定の例のアップデータ・コンポーネントの実行を、更に詳細に説明することにする。このアップデータ・コンポーネントの機能は、「テスト」と呼ばれる導入済みの製品を、すべてのリリースされたパッチを有する全体的に最新の状態に維持することであって、テストの置換バージョンを導入することではない。先ず、アップデータ・コンポーネントが次のようなデータ・インスタンス化によって構成される。

【００９０】

製品ＩＤ：　テスト
現導入済みバージョン：　１．０．ａ
現ライセンス：　ＬＩＣ１
導入環境：　"USERID:TestOwner, USERPASSWORD:easy"
　　"INSTALLPATH: /usr/bin/testapp/"
成長サイクル：　週
成長タイプ：　パッチ、最新、自動的
強制成長：　なし
最終成長時：　０８／１０／９７　月曜日

アップデータは毎週、例えば、各月曜日の夜の午前３時に実行される（タイミングを決定するのはシステム・アドミニストレータである）。

【００９１】以下には、この例のアップデータ・コンポ

ーネントに対する可能な実行トレースが示される。
【００９２】実行トレースの例
ステップ１．　成長サイクルが開始する：
>>>> START : Discover_possible_Growth_Paths()
* フレーズ（"IBM Test 1.0.a Growth Paths"）を使用してリモート・サーチ・エンジン（例えば、インターネット・サーチ・エンジン）におけるサーチを実行する。サーチは、ベンダが製品に対する現在の成長パスを概説することによって公表されたＵＲＬを戻す。
* ＵＲＬをダウンロードする：ファイル内容は：
"1.0.b,none; 2.0, other_required product_product_id 1.0.c;"
* ハッシュ・アルゴリズム及びディジタル・シグニチャを使用してＵＲＬファイルを認証する。真正でない場合、他のＵＲＬ適合基準に関するサーチに戻る。
* growth_path_list を形成する：growth_path list = "1.0.c,none;2.0,other_required_product_id 1.0.c;"
* すべてを除去するが、パッチ・レベルは Growth_path リスト（即ち、第１バージョン及び１．０にマッチした第２リリース番号を有するものだけ）から増加する（Growth_Policy による）。
* growth_path list = "1.0.b,none;"
* リストにおける全メンバに対して、前提が存在することを保証する。この例では、リストのすべてのメンバがこの基準に普通に合致する。
* 候補の growth_paths を Possible_Growth_Paths list = 1.0.b に入れる。
<<<< END : Discover_possible_Growth_Paths()
【００９３】ステップ２．　次に、アップデータ・コンポーネントは成長パスを介して以下の過程をたどる：
>>>> START Decide_Growth_Path()
* 成長ポリシは、最新のパッチされた改訂に我々が成長しなければならないことを指令する。（この例では、最新の改訂は普通であること、即ち、それは１．０．ｂであることを決定する）
* chosen_growth_path = 1.0.b
<<<< END : Decide_Growth_Path()
【００９４】ステップ３．　次に、アップデータ・コンポーネントは、現在のソフトウエア・レベルを新しいソフトウエア・レベルに修正するための必要なリソースを得る。
>>>> Get_Resources()
* フレーズ（"IBM Test REVISION 1.0.a to 1.0.b RESOURCES"）を使用してリモート・サーチ・エンジン（例えば、インターネット・サーチ・エンジン）においてサーチを実行する。
* サーチはＵＲＬ、例えば、次のものを戻す。
ftp://ftp.vendor-site/pub/test/resources/1.0.a-b"
* アップデータはＵＲＬによって指示されたファイルをダウンロードし、それが真偽を検査する機密保持エリアに入れる。
* アップデータは真偽を（例えば、ＲＳＡアルゴリズムに基づくディジタル・シグニチャ、又は他の方法を使用して）検査する。
ファイルが真正でない場合、サーチに戻る（下記の注意１参照）
* アップデータはリソースをアンパックして一時ディレクトリにする（下記の注意２参照）。これらのリソースは機械処理可能な導入命令（例えば、ＵＮＩＸシェル・スクリプト又はＭＶＳ　ＲＥＸＸのようなスクリプト言語で書かれた命令）及び実際にはソフトウエア・フィックスを含むファイル（バイナリ・コンパイル又は要求コンパイル）を含む。
<<<< END : Get_Resources()
【００９５】上記のタスクに関する注意
注意１ － 時間を節約するために、アップデータは「シグニチャ」と呼ばれ、ＵＲＬを含む標準ファイルを、ＵＲＬのダウンロードの前に捜す。
ftp://ftp.vendor-site/pub/test/resources/1.0.a-b
及びその内容のリスト。
これはハッシュされ、サインされる。このシグニチャを使用して、アップデータ・コンポーネントは、（或る範囲までの）ＵＲＬの真偽を、それのダウンロードの前に迅速に確立し、最後のダウンロードされたリソースが一時ディレクトリにアンパックされた後にそれらのリソースを確認するためにその情報、即ち、ファイル・リストを使用することができる。最後のＵＲＬがダウンロードされる時、それは再び真偽をチェックされる（誰かが真正なＵＲＬロケーションに贋造物を配置しないように保護するために）。
注意２ － アンパッキングの部分は、アップデータ・コンポーネントが導入スクリプトを調べ、必要な場合には、それの導入環境データの内容に基づいてそれらを修正することである。例えば、導入命令がシェル・スクリプトでコード化される場合、それは INSTALLPATH のすべてのインスタンスをトークン "/usr/bin/testapp/" でもって置換するであろう。再び、属性のネーミング規約が、導入命令におけるトークン代用の方法であるので標準化される。これは、全体的に自動的導入を可能にする。
【００９６】ステップ４　次に、アップデータ・コンポーネントは実際のソフトウエア・アップグレードを実施する。
>>>> START Install_Resources()
* 導入命令を実行する。
* 次のような値を更新する：
Current_Installed_Version = 1.0.b
Last_Growth_Time = Date+Time
* アップグレードが効果を現す前に、導入と、オペレーティング・システムのリブート又はアプリケーションの

再始動が必要であるかどうかとを知らせる電子メールをソフトウエア・アセット・マネージャに送る。

<<<< END Install_Resources()

【００９７】これはこの現成長サイクルの終了である。シードは現時点で Last_Growth_Time 値を更新し、しかる後、終了する。このサイクルのために費やされる時間は、現在導入されているバージョンに対するアップグレード・パスがないことをアップデータ・コンポーネントが知っている場合の数秒から、現在のものからの全体的に新しいリリースがダウンロードされて新しい前提ソフトウエアと共に導入されることになる場合の数時間までのどれかになり得るであろう。

【００９８】詳細に説明した上記実施例に対する代替は異なる各ソフトウエア製品に対して独立のアップデータ・コンポーネントを必要とせず、各製品と共にダウンロードされる製品特有のプラグ・イン・オブジェクト及び命令と共にシステム上に導入される単一の汎用アップデータ・コンポーネントを使用する。これらのオブジェクトは、汎用コードと相互協調して上記製品特有のアップデータ・コンポーネントの同じ機能を提供する。システム上に導入されたすべてのアプリケーション・プログラムではなく或るアプリケーション・プログラム及び他のソフトウエア製品が関連のアップデータ・コンポーネントを有するというようなシステムにおいて本発明が実施可能であること及び本発明の技術的範囲内で上記実施例に対する別の変更が可能であることは当業者には明らかであろう。

【００９９】まとめとして、本発明の構成に関して以下の事項を開示する。

【０１００】（１）コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入された１つ又は複数のコンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントにして、１つ又は複数の必要なソフトウエア・リソースを検索するために、前記１つ又は複数の必要なソフトウエア・リソースが設けられた前記ネットワーク内の１つ又は複数の識別可能なロケーションへのアクセスを開始するための手段と、１つ又は複数の検索されたソフトウエア・リソースを使用して前記導入されたコンピュータ・プログラムの１つにソフトウエア更新を施すための手段と、を含むアップデータ・コンポーネント。

（２）使用可能な関連の更新リソースを識別するために、前記１つ又は複数の識別可能なロケーションから得られるソフトウエア更新リソースと前記コンピュータ・システム上に導入された１つ又は複数のコンピュータ・プログラムとの比較を行い、前記使用可能な関連の更新リソースと前記コンピュータ・システムにおいて記憶された事前定義の更新基準とを比較するための手段と、ソフトウエア・リソースを自動的にダウンロードし、前記事前定義の更新基準を満たすソフトウエア更新を施すための手段と、を含む上記（１）に記載のアップデータ・コンポーネント。

（３）前記事前定義の更新基準は適用可能なソフトウエア製品ユーザ・ライセンスによる許容し得る更新の範囲の定義を含む上記（２）に記載のアップデータ・コンポーネント。

（４）前記ソフトウエア更新を施すための手段は前記事前定義の更新基準に従って及び更新のためにダウンロードされたソフトウエア・リソースの一部である導入のためのコンピュータ読み取り可能な命令に従って、使用可能な関連のソフトウエア・リソースを導入するための手段を含む上記（２）又は上記（３）に記載のアップデータ・コンポーネント。

（５）１つ又は複数のロケーションを識別するための情報が前記アップデータ・コンポーネントによって保持され、コンピュータ・プログラム製品の製品識別子を含み、前記アップデータ・識別子は前記製品識別子をサーチ・エンジンに供給するように適応し、前記製品識別子は前記サーチ・エンジンがネットワーク・ロケーションを識別するために使用するためのサーチ・パラメータとして働く、上記（１）乃至上記（４）の何れかに記載のアップデータ・コンポーネント。

（６）前記アップデータ・コンポーネントは、使用可能なソフトウエア更新リソースのリストが保持されているネットワーク・ロケーションを前記サーチ・エンジンが識別することに応答して前記リスト及び前記リソースの前提ソフトウエア製品をダウンロードし、前記リスト及び前提ソフトウエア製品と前記コンピュータ・システム上に導入されたコンピュータ・プログラムとを比較し、前記前提ソフトウエア製品に対する更新が必要である場合に前記前提ソフトウエア製品に対する更新をリクエストするように適応する上記（５）に記載のアップデータ・コンポーネント。

（７）前記アップデータ・コンポーネントはコンピュータ・システム上に前記アップデータ・コンポーネントを導入するための機械読み取り可能な導入命令を有し、前記導入命令は前記アップデータ・コンポーネントが他のアップデータ・コンポーネントによって識別可能及び接触可能であるように他のアップデータ・コンポーネントによってアクセスし得るリポジトリによって前記アップデータ・コンポーネントを登録するための命令を含む、上記（１）乃至上記（６）の何れかに記載のアップデータ・コンポーネント。

（８）前記アップデータ・コンポーネントは、現在のアップデータ・コンポーネントがそれのコンピュータ・プログラムを更新することを相補的なコンピュータ・プログラムがリクエストする時に介するＡＰＩを含み、前記現在のアップデータ・コンポーネントは更新リクエストに応答してそれのコンピュータ・プログラムを更新するために更新方法を呼び出すように適応し、前記現在のア

ップデータ・コンポーネントは、それのコンピュータ・プログラムが前提コンピュータ・プログラムの更新を必要とする時、システム発生されたリクエストをそれのコンピュータ・プログラムの前記前提コンピュータ・プログラムのアップデータ・コンポーネントに送るように適応する上記（６）又は上記（７）に記載のアップデータ・コンポーネント。

（９）前記更新を施すための手段は存在する導入済みのソフトウエアを修正する訂正及び機能拡張ソフトウエアを導入し、導入されたソフトウエアを置換する導入済みのソフトウエアのアップグレードしたバージョンを導入するように適応する上記（１）乃至上記（８）の何れかに記載のアップデータ・コンポーネント。

（１０）コンピュータ読み取り可能な記録媒体上に記録されたコンピュータ・プログラム・コードを含み、前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための上記（１乃至上記（９の何れか）に記載された統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

（１１）コンピュータ読み取り可能な記録媒体上のレコーダルのためのコンピュータ・プログラム・コードを含み、前記コンピュータ・プログラム・コードは前記コンピュータ・プログラム・コードを更新するための上記（１）乃至上記（９）の何れかに従って統合アップデータ・コンポーネントを含むコンピュータ・プログラム製品。

（１２）コンピュータ・ネットワーク内で接続されたコンピュータ・システム上に導入されたコンピュータ・プログラムを自動的に更新するための方法にして、前記コンピュータ・プログラムを更新する場合に使用するためのアップデータ・コンポーネントを前記コンピュータ・システムに配布するステップと、前記コンピュータ・プログラムを現在のバージョンから更新済みのバージョンに形成するためのダウンロード可能なソフトウエア・リソースを第１ネットワーク・ロケーションに提供するステップと、前記コンピュータ・システムにおいて実行される時、前記アップデータ・コンポーネントが遂行するように適応するステップであって、（ａ）前記アップデータ・コンポーネントによって保持された情報から識別可能であり、又は前記アップデータ・コンポーネントによってアクセス可能であって、前記ソフトウエア・リソースが配置される前記第１ネットワーク・ロケーションへのアクセスを開始するステップと、（ｂ）前記ソフトウエア・リソースを前記コンピュータ・システム上にダウンロードするステップと、（ｃ）ダウンロードされたソフトウエア・リソースを使用して前記コンピュータ・プログラムを前記現在のバージョンから前記更新済みのバージョンに更新するステップと、を含む方法。

（１３）前記アップデータ・コンポーネントにおける情報から識別可能な第２ネットワーク・ロケーションに前記コンピュータ・プログラムにとって使用可能な更新のコンピュータ読み取り可能なリストを設けるステップを含み、前記アップデータ・コンポーネントによって、前記第１ネットワーク・ロケーションをアクセスする前に遂行されるように適応するステップにして、前記リストを検索するために前記第２ネットワーク・ロケーションへのアクセスを開始するステップと、使用可能な関連の更新リソースを識別するために、前記リストを読み取り、リストされた使用可能な更新と前記第１コンピュータ・システム上の前記コンピュータ・プログラムとの比較を行うステップと、前記使用可能な関連の更新リソースと前記アップデータ・コンポーネントにおける事前定義された更新基準とを比較し、前記更新基準を満たす更新のための使用可能な関連の更新リソースを識別するステップと、を含む上記（１２）に記載の方法。

【図面の簡単な説明】

【図１】導入されたアップデータ・コンポーネントを有するローカル・コンピュータ・システム、使用可能な更新のリスト及び更新を施すためのソフトウエア・リソースを記憶するサーバ・コンピュータ、及びサーバを見つけるためのサーチ・エンジンを含むコンピュータ・ネットワークの概略図である。

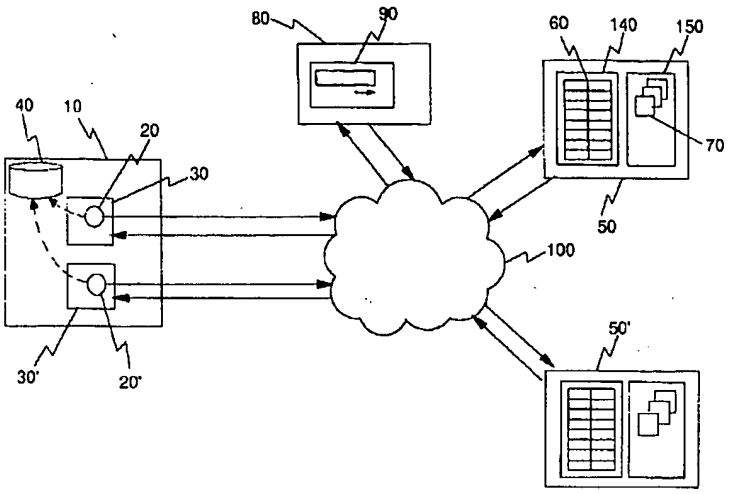
【図２】ソフトウエア・バージョン、及び或るバージョンから別のバージョンを形成するためのリソース及び前提のソフトウエア・ベンダのリストの一例である。

【図３】本発明の実施例に従ってアップデータ・コンポーネントの動作シーケンスを表す。

【図４】アップデータ・コンポーネントのオペレーションのシーケンスの一部分を更に示す。

【図５】アップデータ・コンポーネントのオペレーションのシーケンスの他の部分を更に示す。

【図１】

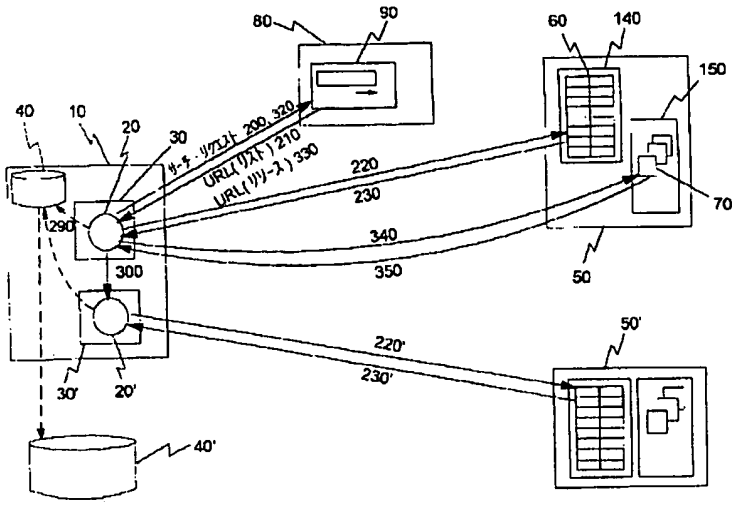


【図２】

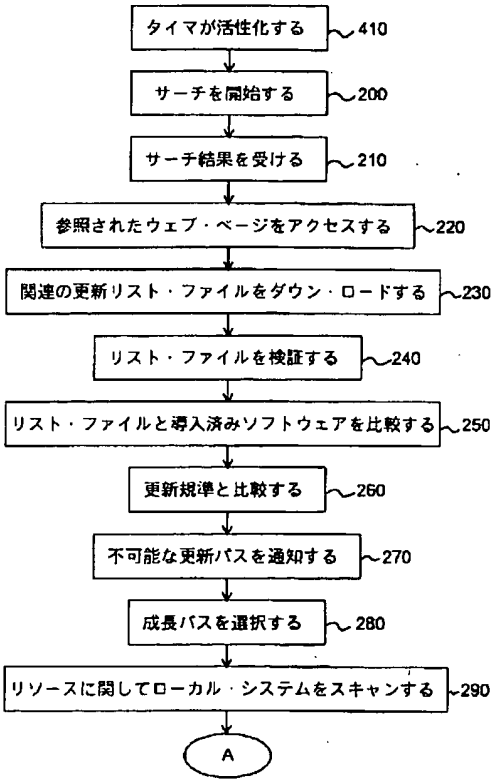
製品セット	更新リソース	前　提
ソフト製品１　v1.0.0	—	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v2.0
ソフト製品１　v1.0.1	ソフト製品１に対するパッチ１	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v2.0
ソフト製品１　v2.0.0	ソフト製品１に対するパッチ２	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v2.0
ソフト製品１　v3.0.0	ソフト製品１　v3.0.0 （置換）	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v2.0
ソフトゲーム１　v1.0	——	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v2.0
ソフトゲーム２　v2.0	ソフトゲーム２に対するパッチ１	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v3.0
ソフトゲーム３　v3.0	ソフトゲーム２に対するパッチ２	ｵﾍﾟﾚｰﾃｨﾝｸﾞ・ｼｽﾃﾑ T3　v3.0

120　60　110　130　160

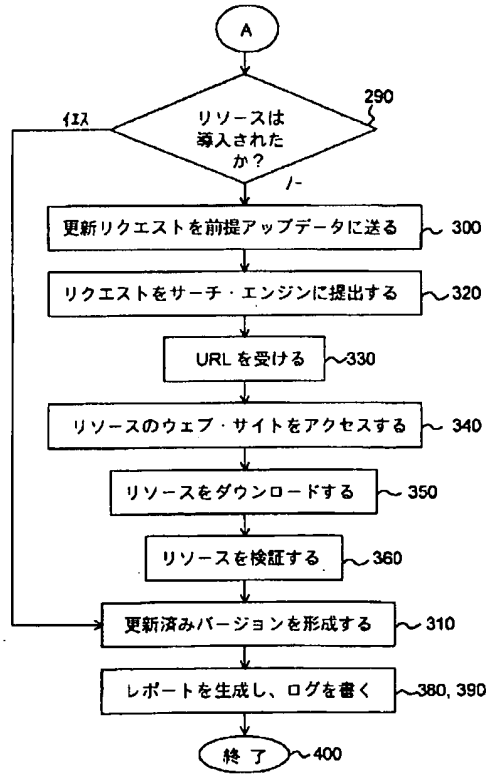
【図３】


80
90
60
140
150
40
10
20
30
サーチ・リクエスト 200, 320
URL(リスト) 210
URL(リソース) 330
220
230
70
290
340
350
300
50
50'
220'
230'
30'
20'
40'

【図４】


タイマが活性化する
410
サーチを開始する
200
サーチ結果を受ける
210
参照されたウェブ・ページをアクセスする
220
関連の更新リスト・ファイルをダウン・ロードする
230
リスト・ファイルを検証する
240
リスト・ファイルと導入済みソフトウェアを比較する
250
更新規準と比較する
260
不可能な更新パスを通知する
270
成長パスを選択する
280
リソースに関してローカル・システムをスキャンする
290
A

【図５】


A
リソースは導入されたか？
290
イエス
ノー
更新リクエストを前提アップデータに送る
300
リクエストをサーチ・エンジンに提出する
320
URL を受ける
330
リソースのウェブ・サイトをアクセスする
340
リソースをダウンロードする
350
リソースを検証する
360
更新済みバージョンを形成する
310
レポートを生成し、ログを書く
380, 390
終了
400